

Overview and applications of the SSAHA2 package.

Adam Spargo and Zemin Ning.

High performance assembly group.

Wellcome Trust Sanger Institute.

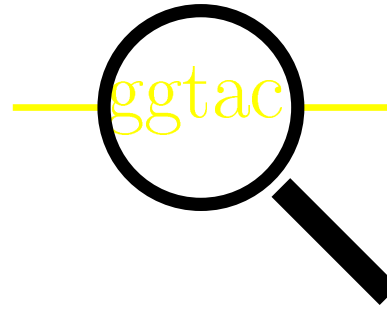
Hinxton, Cambridge, UK.

07/08/2006

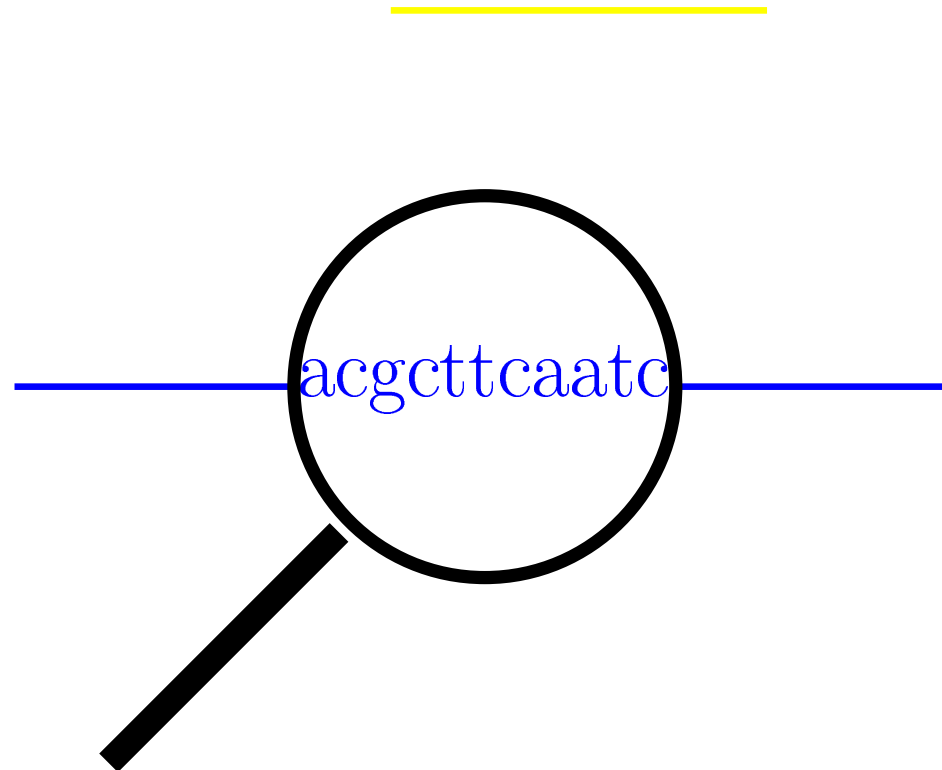
- ⇒⇒ Sequence Alignment Overview.
- ⇒⇒ The SSAHA Algorithm.
- ⇒⇒ The SSAHA2 Package - installation from download.
- ⇒⇒ SSAHA2 options.
- ⇒⇒ Running large jobs.
- ⇒⇒ ssaha2Server.
- ⇒⇒ TraceSearch.
- ⇒⇒ QUESTIONS.
- ⇒⇒ cross_genome - Aligning whole genomes.
- ⇒⇒ ssahaEST - handling splice sites correctly.
- ⇒⇒ ssahaSNP - detecting SNPs and short indels.
- ⇒⇒ ssahaSV - detecting structural variations.
- ⇒⇒ QUESTIONS.



We want to compare two sequences, which we refer to as the query Q and subject S .



The query is usually a short sequence such as a wgs read, primer or EST.



The subject is usually a long sequence such as a finished genome or a database of short sequences.

J. Mol. Biol. (1981), **147**, 195–197

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of “events” required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of “mutational events” required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_m$. A similarity $s(a, b)$ is given between sequence elements a and b . Deletions of length k are given weight W_k . To find pairs of segments with high degrees of similarity, we set up a matrix H . First set

$$H_{k0} = H_{0l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m.$$

Preliminary values of H have the interpretation that H_{ij} is the maximum similarity of two segments ending in a_i and b_j , respectively. These values are obtained from the relationship

$$H_{ij} = \max \{ H_{i-1, j-1} + s(a_i, b_j), \max_{k \geq 1} \{ H_{i-k, j} - W_k \}, \max_{l \geq 1} \{ H_{i, j-l} - W_l \}, 0 \}, \quad (1)$$

$1 \leq i \leq n$ and $1 \leq j \leq m$.

196

0022-2836/80/000195-03 \$02.00/0

© 1980 Academic Press Inc. (London) Ltd.

J. Mol. Biol. (1981), **147**, 195–197

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of “events” required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of “mutational events” required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_m$. A similarity $s(a, b)$ is given between sequence elements a and b . Deletions of length k are given weight W_k . To find pairs of segments with high degrees of similarity, we set up a matrix H . First set

$$H_{k0} = H_{0l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m.$$

Preliminary values of H have the interpretation that H_{ij} is the maximum similarity of two segments ending in a_i and b_j , respectively. These values are obtained from the relationship

$$H_{ij} = \max \{ H_{i-1, j-1} + s(a_i, b_j), \max_{k \geq 1} \{ H_{i-k, j} - W_k \}, \max_{l \geq 1} \{ H_{i, j-l} - W_l \}, 0 \}, \quad (1)$$

$1 \leq i \leq n$ and $1 \leq j \leq m$.

196

0022-2836/80/000195-03 \$02.00/0

© 1980 Academic Press Inc. (London) Ltd.

Given two sequences Q and S , with
 $|Q| = n$ and $|S| = m$

J. Mol. Biol. (1981), **147**, 195–197

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of “events” required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of “mutational events” required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_m$. A similarity $s(a, b)$ is given between sequence elements a and b . Deletions of length k are given weight W_k . To find pairs of segments with high degrees of similarity, we set up a matrix H . First set

$$H_{k0} = H_{0l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m.$$

Preliminary values of H have the interpretation that H_{ij} is the maximum similarity of two segments ending in a_i and b_j , respectively. These values are obtained from the relationship

$$H_{ij} = \max \{ H_{i-1, j-1} + s(a_i, b_j), \max_{k \geq 1} \{ H_{i-k, j} - W_k \}, \max_{l \geq 1} \{ H_{i, j-l} - W_l \}, 0 \}, \quad (1)$$

$1 \leq i \leq n$ and $1 \leq j \leq m$.

196

0022-2836/80/000195-03 \$02.00/0

© 1980 Academic Press Inc. (London) Ltd.

Given two sequences Q and S , with $|Q| = n$ and $|S| = m$, construct the $n \times m$ matrix $M = [M_{i,j}]$

J. Mol. Biol. (1981), **147**, 195–197

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of “events” required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of “mutational events” required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_m$. A similarity $s(a, b)$ is given between sequence elements a and b . Deletions of length k are given weight W_k . To find pairs of segments with high degrees of similarity, we set up a matrix H . First set

$$H_{k0} = H_{0l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m.$$

Preliminary values of H have the interpretation that H_{ij} is the maximum similarity of two segments ending in a_i and b_j , respectively. These values are obtained from the relationship

$$H_{ij} = \max \{ H_{i-1, j-1} + s(a_i, b_j), \max_{k \geq 1} \{ H_{i-k, j} - W_k \}, \max_{l \geq 1} \{ H_{i, j-l} - W_l \}, 0 \}, \quad (1)$$

$1 \leq i \leq n$ and $1 \leq j \leq m$.

196

0022-2836/80/000195-03 \$02.00/0

© 1980 Academic Press Inc. (London) Ltd.

Given two sequences Q and S , with $|Q| = n$ and $|S| = m$, construct the $n \times m$ matrix $M = [M_{i,j}]$ such that

$$M_{i,j} = \max \left\{ \begin{array}{l} 0, \\ M_{i-1,j-1} + \delta(Q_i, S_j), \\ M_{i,j-1} - d, \\ M_{i-1,j} - d \end{array} \right\}$$

where i runs from 1 to n , j runs from 1 to m , $\delta(., .)$ is a scoring function and d is the gap-penalty.

J. Mol. Biol. (1981), **147**, 195–197

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of “events” required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of “mutational events” required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_m$. A similarity $s(a, b)$ is given between sequence elements a and b . Deletions of length k are given weight W_k . To find pairs of segments with high degrees of similarity, we set up a matrix H . First set

$$H_{k0} = H_{0l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m.$$

Preliminary values of H have the interpretation that H_{ij} is the maximum similarity of two segments ending in a_i and b_j , respectively. These values are obtained from the relationship

$$H_{ij} = \max \{ H_{i-1, j-1} + s(a_i, b_j), \max_{k \geq 1} \{ H_{i-k, j} - W_k \}, \max_{l \geq 1} \{ H_{i, j-l} - W_l \}, 0 \}, \quad (1)$$

$1 \leq i \leq n$ and $1 \leq j \leq m$.

196

0022-2836/80/000195-03 \$02.00/0

© 1980 Academic Press Inc. (London) Ltd.

Given two sequences Q and S , with $|Q| = n$ and $|S| = m$, construct the $n \times m$ matrix $M = [M_{i,j}]$ such that

$$M_{i,j} = \max \left\{ \begin{array}{l} 0, \\ M_{i-1,j-1} + \delta(Q_i, S_j), \\ M_{i,j-1} - d, \\ M_{i-1,j} - d \end{array} \right\}$$

where i runs from 1 to n , j runs from 1 to m , $\delta(., .)$ is a scoring function and d is the gap-penalty.

$$M_{i,0} = 0; \quad i = 0, \dots, n.$$

$$M_{0,j} = 0; \quad j = 0, \dots, m.$$

The optimal alignment between Q and S ends at Q_u, S_v for u, v such that $M_{u,v}$ is the largest element of \mathbf{M} . This alignment is said to have score $M_{u,v}$.

The optimal alignment between Q and S ends at Q_u, S_v for u, v such that $M_{u,v}$ is the largest element of \mathbf{M} . This alignment is said to have score $M_{u,v}$.

Therefore to know the score of the optimal alignment between two sequences for given $\delta(., .)$ and d , we simply need to fill in the matrix \mathbf{M} and keep a record of the maximum entry as we go.

The optimal alignment between Q and S ends at Q_u, S_v for u, v such that $M_{u,v}$ is the largest element of \mathbf{M} . This alignment is said to have score $M_{u,v}$.

Therefore to know the score of the optimal alignment between two sequences for given $\delta(., .)$ and d , we simply need to fill in the matrix \mathbf{M} and keep a record of the maximum entry as we go.

Example:

$Q = AACATTG GTGATTG$

$S = TGATTGAGATTGATGC$

$$\delta(x, y) = \begin{cases} 1 & x = y \\ -2 & x \neq y \end{cases}$$

$$d = 1.$$

Filling in the first row we have

$$\begin{aligned} M_{1,1} &= \max \left\{ \begin{array}{l} 0, \\ M_{0,0} + \delta(A, T), \\ M_{1,0} - d, \\ M_{0,1} - d \end{array} \right\} \\ &= \max \left\{ \begin{array}{l} 0, \\ 0 - 2, \\ 0 - 1, \\ 0 - 1 \end{array} \right\} \\ &= \max \{0, -2, -1, -1\} \\ &= 0. \end{aligned}$$

Filling in the first row we have

similarly

$$\begin{aligned}
 M_{1,1} &= \max \left\{ \begin{array}{l} 0, \\ M_{0,0} + \delta(A, T), \\ M_{1,0} - d, \\ M_{0,1} - d \end{array} \right\} \\
 &= \max \left\{ \begin{array}{l} 0, \\ 0 - 2, \\ 0 - 1, \\ 0 - 1 \end{array} \right\} \\
 &= \max \{0, -2, -1, -1\} \\
 &= 0.
 \end{aligned}
 \qquad
 \begin{aligned}
 M_{1,2} &= \max \left\{ \begin{array}{l} 0, \\ M_{0,1} - 2, \\ M_{1,1} - 1, \\ M_{0,2} - 1 \end{array} \right\} \\
 &= \max \{0, -2, -1, -1\} \\
 &= 0.
 \end{aligned}$$

and

$$\begin{aligned} M_{1,3} &= \max \left\{ \begin{array}{l} 0, \\ M_{0,2} + 1, \\ M_{1,2} - 1, \\ M_{0,3} - 1 \end{array} \right\} \\ &= \max \{0, 1, -1, -1\} \\ &= 1. \end{aligned}$$

and

$$\begin{aligned}
 M_{1,3} &= \max \left\{ \begin{array}{l} 0, \\ M_{0,2} + 1, \\ M_{1,2} - 1, \\ M_{0,3} - 1 \end{array} \right\} \\
 &= \max \{0, 1, -1, -1\} \\
 &= 1.
 \end{aligned}$$

and so on

.	<i>T</i>	<i>G</i>	<i>A</i>	<i>T</i>	<i>T</i>	<i>T</i>	...
<i>A</i>	0	0	1	0	0	0	...
<i>A</i>							
⋮							

Proceeding row by row the full matrix looks like this

·	T	G	A	T	T	T	G	A	G	A	T	T	G	A	T	G	C
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
T	1	0	0	2	1	1	0	0	0	0	2	1	0	0	2	0	0
T	1	0	0	1	3	2	1	0	0	0	1	3	2	1	1	0	0
T	1	0	0	1	2	4	3	2	1	0	1	2	1	0	2	1	0
G	0	2	1	0	1	3	5	4	3	2	1	0	3	2	1	3	2
G	0	1	0	0	0	2	4	3	5	4	3	2	2	1	0	2	1
T	1	0	0	1	1	1	3	2	4	3	5	4	3	2	2	1	0
G	0	2	1	0	0	0	2	1	3	2	4	3	5	4	3	3	2
A	0	1	3	2	1	0	1	3	2	4	3	2	4	6	5	4	3
T	1	0	2	4	3	2	1	2	1	3	5	4	3	5	7	6	5
T	1	0	0	3	5	4	3	2	1	2	4	6	5	4	6	5	4
G	0	2	1	2	4	3	5	4	3	2	3	5	7	6	5	7	6

Trace back to find the alignments

.	T	G	A	T	T	T	G	A	G	A	T	T	G	A	T	G	C
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
T	1	0	0	2	1	1	0	0	0	0	2	1	0	0	2	0	0
T	1	0	0	1	3	2	1	0	0	0	1	3	2	1	1	0	0
T	1	0	0	1	2	4	3	2	1	0	1	2	1	0	2	1	0
G	0	2	1	0	1	3	5	4	3	2	1	0	3	2	1	3	2
G	0	1	0	0	0	2	4	3	5	4	3	2	2	1	0	2	1
T	1	0	0	1	1	1	3	2	4	3	5	4	3	2	2	1	0
G	0	2	1	0	0	0	2	1	3	2	4	3	5	4	3	3	2
A	0	1	3	2	1	0	1	3	2	4	3	2	4	6	5	4	3
T	1	0	2	4	3	2	1	2	1	3	5	4	3	5	7	6	5
T	1	0	0	3	5	4	3	2	1	2	4	6	5	4	6	5	4
G	0	2	1	2	4	3	5	4	3	2	3	5	7	6	5	7	6

subject 3 ATTT_G_AGATTG 13
 ||||-|--||||

query 4 ATTTGGT_GATTG 15

subject 3 ATTTGA_GATTG 13
 ||||i-||||

query 4 ATTTGGTGATTG 15

subject 3 ATTT_GA_GATTG 13
 ||||-|--||||

query 4 ATTTGG_TGATTG 15

subject 3 ATTTGAGATTGAT 15
 ||||-|-|-|||

query 4 ATTTG_G_T_GAT 13

subject 3 ATTTGAGATTGAT_G 16
 ||||-|-|-|||-|

query 4 ATTTG_G_T_GATTG 15

subject 3 ATTTGAGATTGA_TG 16
 ||||-|-|-|||-|

query 4 ATTTG_G_T_GATTG 15

subject 3 ATTTG__AGATTG 13
 |||||__||||

query 4 ATTTGGT_GATTG 15

subject 3 ATTTGA__GATTG 13
 |||||---||||

query 4 ATTTG_GTGATTG 15

subject 3 ATTTGAG__ATTG 13
 ||||-|--|||

query 4 ATTTG_GTGATTG 15

subject 3 ATTTGAGATTCAT 15
 ||||-|--|||

query 4 ATTTG_G__TGAT 13

subject 3 ATTTGAGATTGAT_G 16
 ||||-|--|||-|

query 4 ATTTG_G__TGATTG 15

subject 3 ATTTGAGATTGA_TG 16
 ||||-|--|||-|

query 4 ATTTG_G__TGATTG 15

- ✓ Always finds optimal alignment(s).

- ✓ Always finds optimal alignment(s).
- ✓ Finds N -best alignments.

- ✓ Always finds optimal alignment(s).
- ✓ Finds N -best alignments.
- ✓ Easy to code.

- ✓ Always finds optimal alignment(s).
 - ✓ Finds N -best alignments.
 - ✓ Easy to code.
- ✗ $O(n^2)$ space.

- ✓ Always finds optimal alignment(s).
- ✓ Finds N -best alignments.
- ✓ Easy to code.
- ✗ $O(n^2)$ space.
- ✗ $O(n^2)$ time.

High scoring alignments are all located along diagonals.

·	T	G	A	T	T	T	G	A	G	A	T	T	G	A	T	G	C
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
A	0	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	0
T	1	0	0	2	1	1	0	0	0	0	2	1	0	0	2	0	0
T	1	0	0	1	3	2	1	0	0	0	1	3	2	1	1	0	0
T	1	0	0	1	2	4	3	2	1	0	1	2	1	0	2	1	0
G	0	2	1	0	1	3	5	4	3	2	1	0	3	2	1	3	2
G	0	1	0	0	0	2	4	3	5	4	3	2	2	1	0	2	1
T	1	0	0	1	1	1	3	2	4	3	5	4	3	2	2	1	0
G	0	2	1	0	0	0	2	1	3	2	4	3	5	4	3	3	2
A	0	1	3	2	1	0	1	3	2	4	3	2	4	6	5	4	3
T	1	0	2	4	3	2	1	2	1	3	5	4	3	5	7	6	5
T	1	0	0	3	5	4	3	2	1	2	4	6	5	4	6	5	4
G	0	2	1	2	4	3	5	4	3	2	3	5	7	6	5	7	6

Proc. Natl. Acad. Sci. USA
Vol. 85, pp. 2444-2448, April 1988
Biochemistry

Improved tools for biological sequence comparison

(amino acid/nucleic acid/data base searches/local similarity)

WILLIAM R. PEARSON* AND DAVID J. LIPMAN†

*Department of Biochemistry, University of Virginia, Charlottesville, VA 22908; and †Mathematical Research Branch, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, MD 20892

Communicated by Gerald M. Rubin, December 2, 1987 (received for review September 17, 1987)

ABSTRACT We have developed three computer programs for comparisons of protein and DNA sequences. They can be used to search sequence data bases, evaluate similarity scores, and identify periodic structures based on local sequence similarity. The FASTA program is a more sensitive derivative of the FASTP program, which can be used to search protein or DNA sequence data bases and can compare a protein sequence to a DNA sequence data base by translating the DNA data base as it is searched. FASTA includes an additional step in the calculation of the initial pairwise similarity score that allows multiple regions of similarity to be joined to increase the score of related sequences. The RDP2 program can be used to evaluate the significance of similarity scores using a shuffling method that preserves local sequence composition. The LFASTA program can display all the regions of local similarity between two sequences with scores greater than a threshold, using the same scoring parameters and a similar alignment algorithm; these local similarities can be displayed as a "graphic matrix" plot or as individual alignments. In addition, these programs have been generalized to allow comparison of DNA or protein sequences based on a variety of alternative scoring matrices.

We have been developing tools for the analysis of protein and DNA sequence similarity that achieve a balance of sensitivity and selectivity on the one hand and speed and memory requirements on the other. Three years ago, we described the FASTP program for searching amino acid sequence data bases (1), which uses a rapid technique for finding identities shared between two sequences and exploits the biological constraints on molecular evolution. FASTP has decreased the time required to search the National Biomedical Research Foundation (NBRF) protein sequence data base by more than two orders of magnitude and has been used by many investigators to find biologically significant similarities to newly sequenced proteins. There is a trade-off between sensitivity and selectivity in biological sequence comparison: methods that can detect more distantly related sequences (increased sensitivity) frequently increase the similarity scores of unrelated sequences (decreased selectivity). In this paper we describe a new version of FASTP, FASTA, which uses an improved algorithm that increases sensitivity with a small loss of selectivity and a negligible decrease in speed. We have also developed a related program, LFASTA, for local similarity analyses of DNA or amino acid sequences. These programs run on commonly available microcomputers as well as on larger machines.

METHODS

The search algorithm we have developed proceeds through four steps in determining a score for pair-wise similarity.

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "advertisement" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

FASTP and FASTA achieve much of their speed and selectivity in the first step, by using a lookup table to locate all identities or groups of identities between two DNA or amino acid sequences during the first step of the comparison (2). The *k*rup parameter determines how many consecutive identities are required in a match. For example, if *k*rup = 4 for a DNA sequence comparison, only those identities that occur in a run of four consecutive matches are examined. In the first step, the 10 best diagonal regions are found using a simple formula based on the number of *k*rup matches and the distance between the matches without considering shorter runs of identities, conservative replacements, insertions, or deletions (1, 3).

In the second step of the comparison, we rescore these 10 regions using a scoring matrix that allows conservative replacements and runs of identities shorter than *k*rup to contribute to the similarity score. For protein sequences, this score is usually calculated using the PAM250 matrix (4), although scoring matrices based on the minimum number of base changes required for a replacement or on an alternative measure of similarity can also be used with FASTA. For each of these best diagonal regions, a subregion with maximal score is identified. We will refer to this region as the "initial region"; the best initial regions from Fig. 1A are shown in Fig. 1B.

The FASTP program uses the single best scoring initial region to characterize pair-wise similarity; the initial scores are used to rank the library sequences. FASTA goes one step further during a library search; it checks to see whether several initial regions may be joined together. Given the locations of the initial regions, their respective scores, and a "joining" penalty (analogous to a gap penalty), FASTA calculates an optimal alignment of initial regions as a combination of compatible regions with maximal score. FASTA uses the resulting score to rank the library sequences. We limit the degradation of selectivity by including in the optimization step only those initial regions whose scores are above a threshold. This process can be seen by comparing Fig. 1B with Fig. 1C. Fig. 1B shows the 10 highest scoring initial regions after rescoring with the PAM250 matrix; the best initial region reported by FASTP is marked with an asterisk. Fig. 1C shows an optimal subset of initial regions that can be joined to form a single alignment.

In the fourth step of the comparison, the highest scoring library sequences are aligned using a modification of the optimization method described by Needleman and Wunsch (5) and Smith and Waterman (6). This final comparison considers all possible alignments of the query and library sequence that fall within a band centered around the highest scoring initial region (Fig. 1D). With the FASTP program, optimization frequently improved the similarity scores of related sequences by factors of 2 or 3. Because FASTA calculates an initial similarity score based on an optimization of initial regions during the library search, the initial score is

Abbreviation: NBRF, National Biomedical Research Foundation.

$k = 5$

·	T	G	A	T	T	T	G	A	G	A	T	T	G	A	T	G	C
A			*					*		*				*			
A			*					*		*				*			
C																	*
A			*					*		*				*			
T	*			*	*	*					*	*			*		
T	*			*	*	*					*	*			*		
T	*			*	*	*					*	*			*		
G		*					*		*				*			*	
G		*					*		*				*			*	
T	*			*	*	*					*	*			*		
G		*					*		*				*			*	
A			*				*		*				*		*		
T	*			*	*	*					*	*			*		
T	*			*	*	*					*	*			*		
G		*					*		*				*			*	

$b = 2$

·	T	G	A	T	T	T	G	A	G	A	T	T	G	A	T	G	C
A	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0
A	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0
T	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0
T	0	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0
T	0	0	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0
G	*	0	0	*	*	*	*	*	*	0	0	0	0	0	0	0	0
G	*	*	0	0	*	*	*	*	*	*	0	0	0	0	0	0	0
T	*	*	*	0	0	*	*	*	*	*	*	0	0	0	0	0	0
G	*	*	*	*	0	0	*	*	*	*	*	*	0	0	0	0	0
A	*	*	*	*	*	0	0	*	*	*	*	*	*	0	0	0	0
T	0	*	*	*	*	*	0	0	*	*	*	*	*	*	0	0	0
T	0	0	*	*	*	*	*	0	0	*	*	*	*	*	*	0	0
G	0	0	0	*	*	*	*	*	0	0	*	*	*	*	*	*	0

J. Mol. Biol. (1990) 215, 403–410

Basic Local Alignment Search Tool

Stephen F. Altschul¹, Warren Gish¹, Webb Miller²
Eugene W. Myers³ and David J. Lipman¹

¹*National Center for Biotechnology Information
National Library of Medicine, National Institutes of Health
Bethesda, M D 20894 USA*

²*Department of Computer Science
The Pennsylvania State University, University Park, PA 16802
U.S.A.*

³*Department of Computer Science
University of Arizona, Tucson, AZ 85721, U.S.A*

(Received 26 February 1990; accepted 15 May 1990)

A new approach to rapid sequence comparison, basic local alignment search tool (BLAST), directly approximates alignments that optimize a measure of local similarity, the maximal segment pair (MSP) score. Recent mathematical results on the stochastic properties of MSP scores allow an analysis of the performance of this method as well as the statistical significance of alignments it generates. The basic algorithm is simple and robust; it can be implemented in a number of ways and applied in a variety of contexts including straight-forward DNA and protein sequence database searches, motif searches, gene identification searches, and in the analysis of multiple regions of similarity in long DNA sequences. In addition to its flexibility and tractability to mathematical analysis, BLAST is an order of magnitude faster than existing sequence comparison tools of comparable sensitivity.

1. Introduction

The discovery of sequence homology to a known protein or family of proteins often provides the first clues about the function of a newly sequenced gene. As the DNA and amino acid sequence databases continue to grow in size they become increasingly useful in the analysis of newly sequenced genes and proteins because of the greater chance of finding such homologies. There are a number of software tools for searching sequence databases but all use some measure of similarity between sequences to distinguish biologically significant relationships from chance similarities. Perhaps the best studied measures are those used in conjunction with variations of the dynamic programming algorithm (Needleman & Wunsch, 1970; Sellers, 1974; Sankoff & Kruskal, 1983; Waterman, 1984). These methods assign scores to insertions, deletions and replacements, and compute an alignment of two sequences that corresponds to the least costly set of such mutations. Such an alignment may be thought of as minimizing the evolutionary distance or maximizing the similarity between the two sequences compared. In either case, the cost of this alignment is a measure of similarity; the algorithm guarantees it is

optimal, based on the given scores. Because of their computational requirements, dynamic programming algorithms are impractical for searching large databases without the use of a supercomputer (Gotoh & Tagashira, 1986) or other special purpose hardware (Coulson *et al.*, 1987).

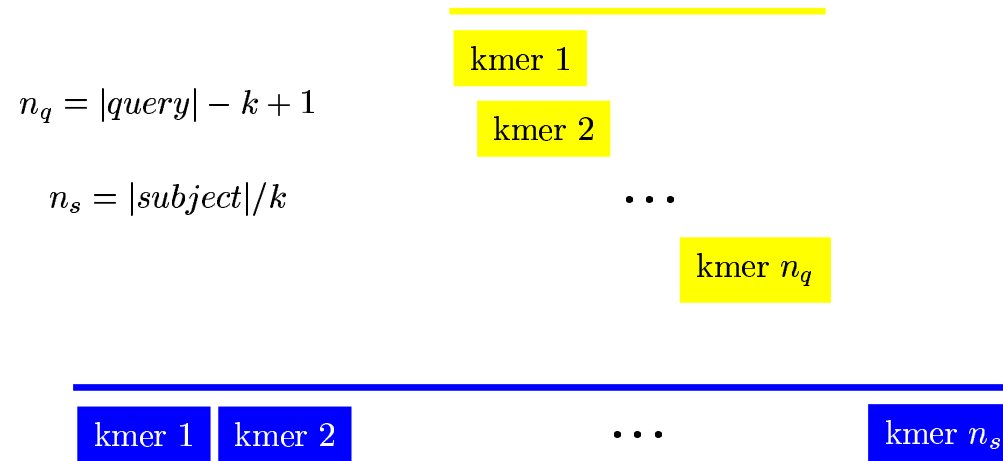
Rapid heuristic algorithms that attempt to approximate the above methods have been developed (Waterman, 1984), allowing large databases to be searched on commonly available computers. In many heuristic methods the measure of similarity is not explicitly defined as a minimal cost set of mutations, but instead is implicit in the algorithm itself. For example, the FASTP program (Lipman & Pearson, 1985; Pearson & Lipman, 1988) first finds locally similar regions between two sequences based on identities but not gaps, and then rescores these regions using a measure of similarity between residues, such as a PAM matrix (Dayhoff *et al.*, 1978) which allows conservative replacements as well as identities to increment the similarity score. Despite their rather indirect approximation of minimal evolution measures, heuristic tools such as FASTP have been quite popular and have identified many distant but biologically significant relationships.

- ⇒ Look for at least two k -mer words on same diagonal.
- ⇒ Extend with ungapped alignment.
- ⇒ Join into gapped alignment.

0022-2833/90/190403-08 \$03.00/0

403

© 1990 Academic Press Limited



Split the subject into non-overlapping k -mers. Process the query in overlapping k -mers.

The algorithm constructs a hashtable consisting of a *head* and a *body*. The *head* contains one integer for every possible k -mer over the alphabet; 4^k for DNA. The *body* contains one integer for every whole k -mer in the subject sequence or database; with $k = 12$ and 4 byte integers, the *body* size is one third the number of bases in the subject.

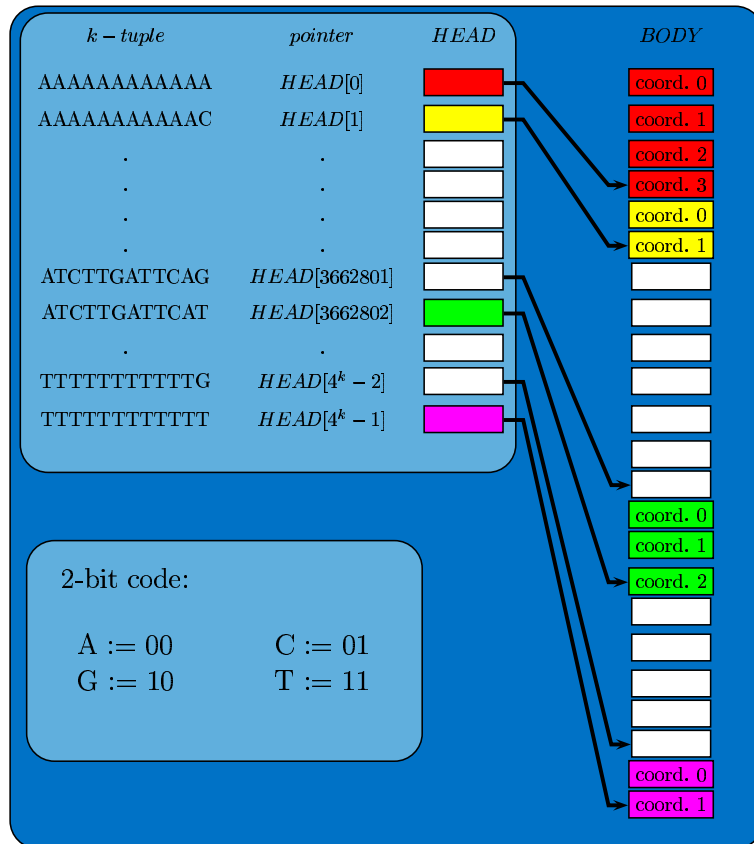
The basis of the algorithm is a 2-bit encoding α^{binary} for each base α of the DNA sequence

$$\alpha^{binary} = \begin{cases} 00 & \text{if } \alpha = A; \\ 01 & \text{if } \alpha = C; \\ 10 & \text{if } \alpha = G; \\ 11 & \text{if } \alpha = T. \end{cases}$$

An integer κ is then constructed for each k -mer by taking the decimal representation of the binary number κ^{binary} obtained by concatenating the 2-bit codes α_i^{binary} for each base α_i of the k -mer in turn

$$\begin{aligned} \kappa^{binary} &= \sum_{i=1}^k \alpha_i^{binary} \cdot (100)_2^{(k-i)} \\ \kappa &= \text{decimal}(\kappa^{binary}). \end{aligned}$$

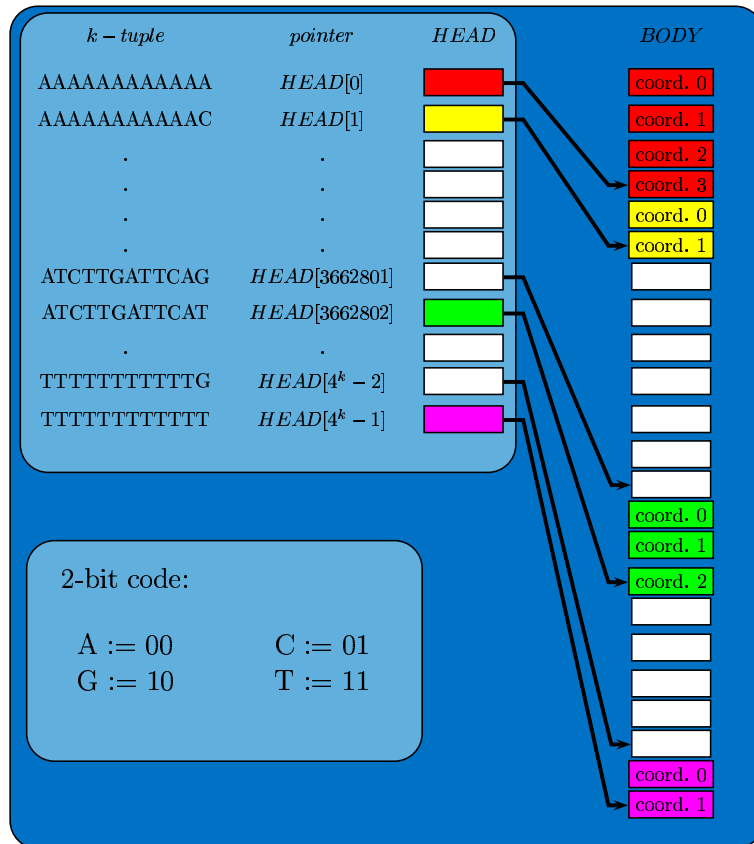
The value of κ for each k -mer is straight forward to construct in practice using bit-shift operations on an integer



- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAG

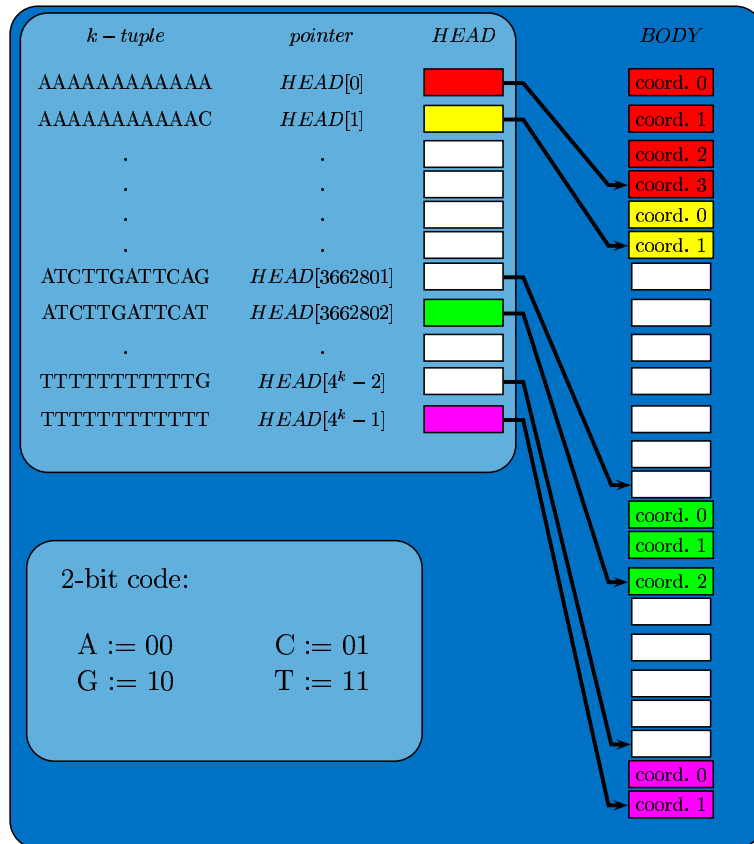




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

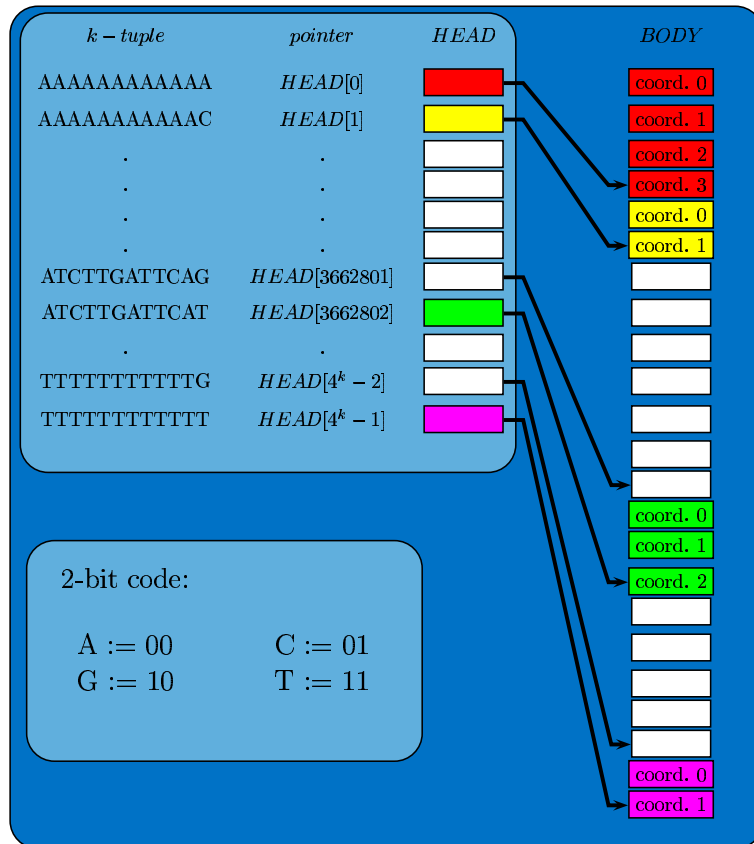




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

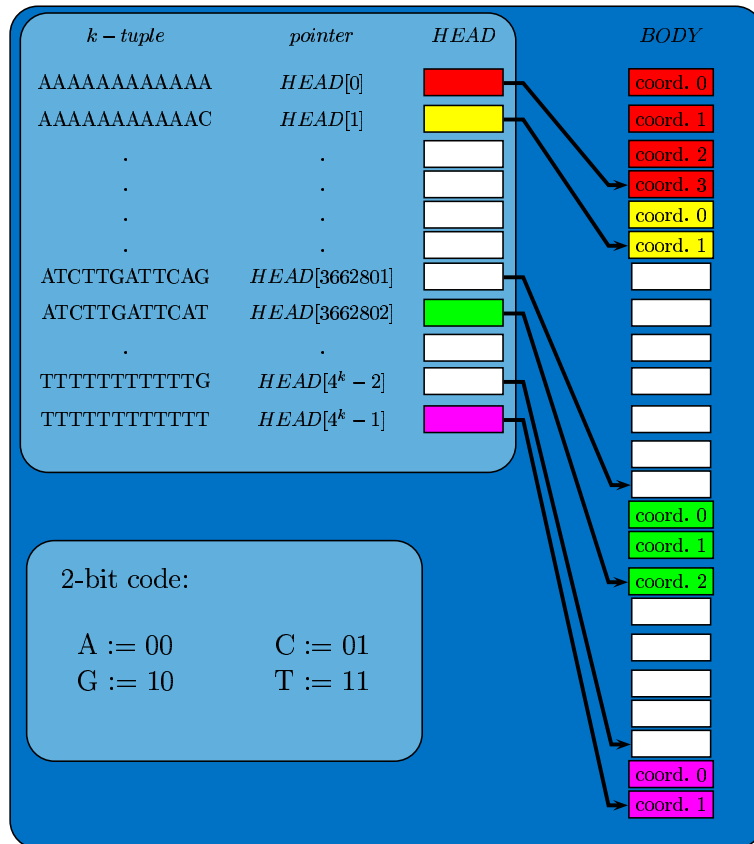




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT





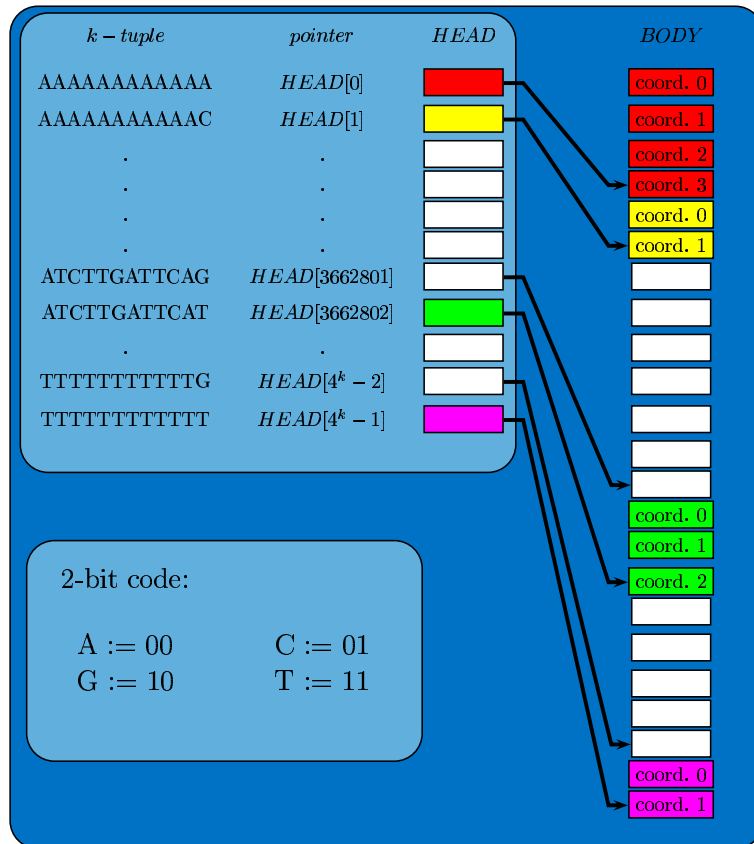
⇒ Notice that DNA has an alphabet of only 4 characters.

⇒ Can encode each base in two binary digits.

⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

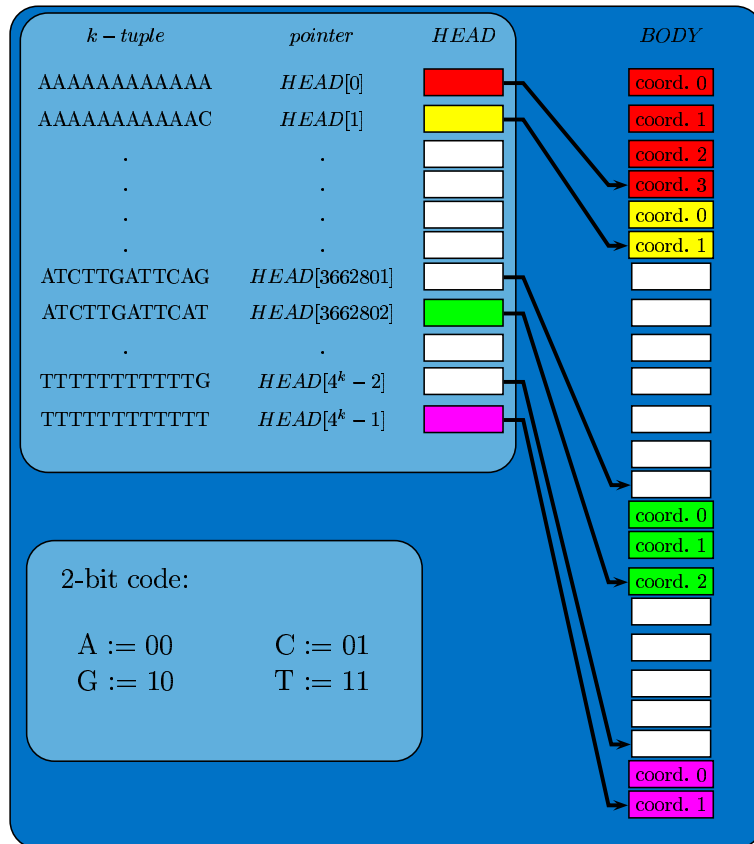




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

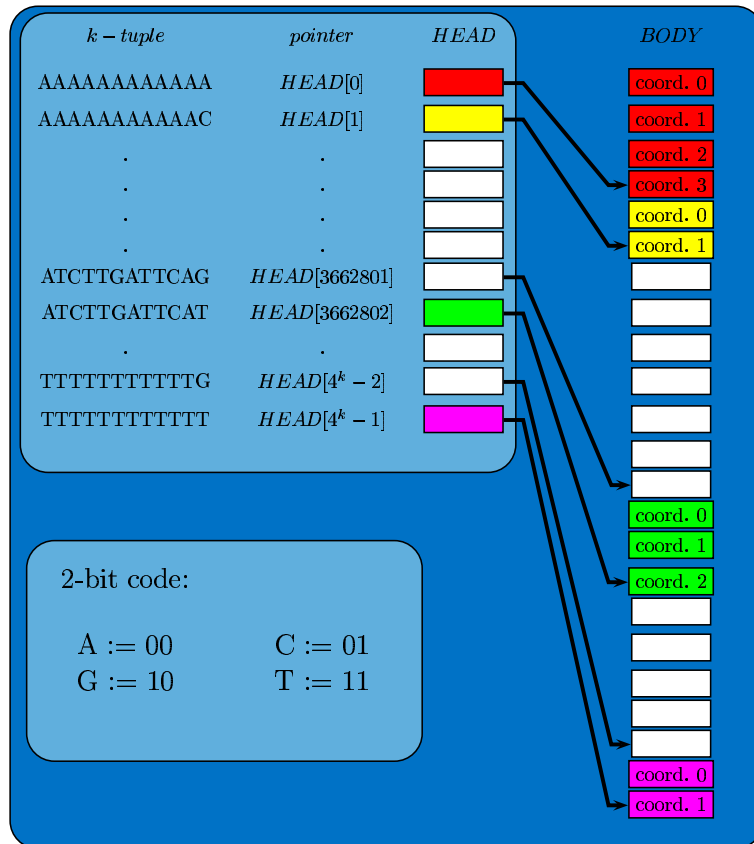




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

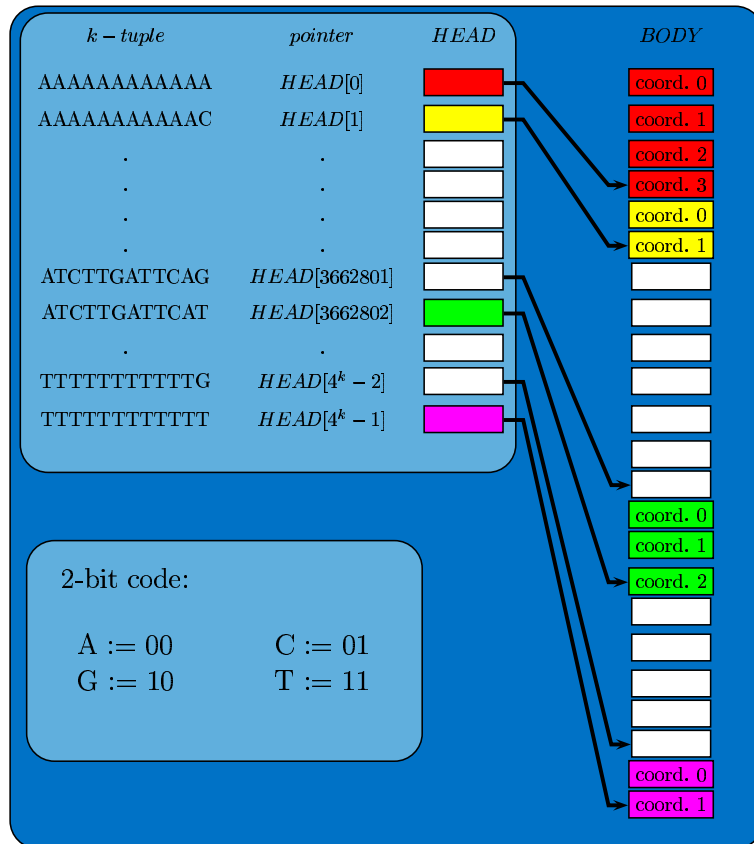




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

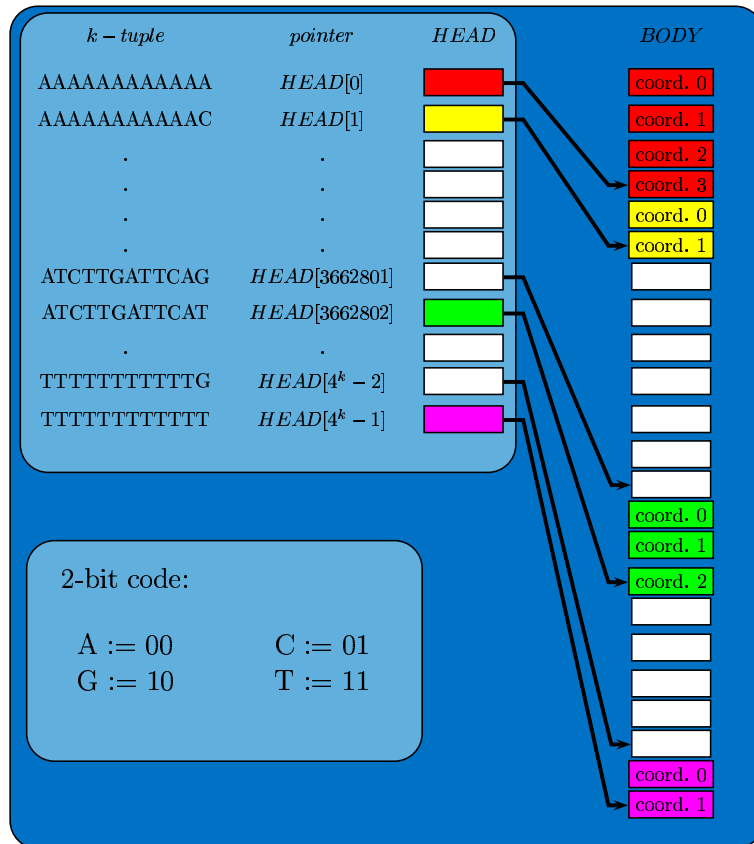
								00	11	01	11
--	--	--	--	--	--	--	--	----	----	----	----



- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

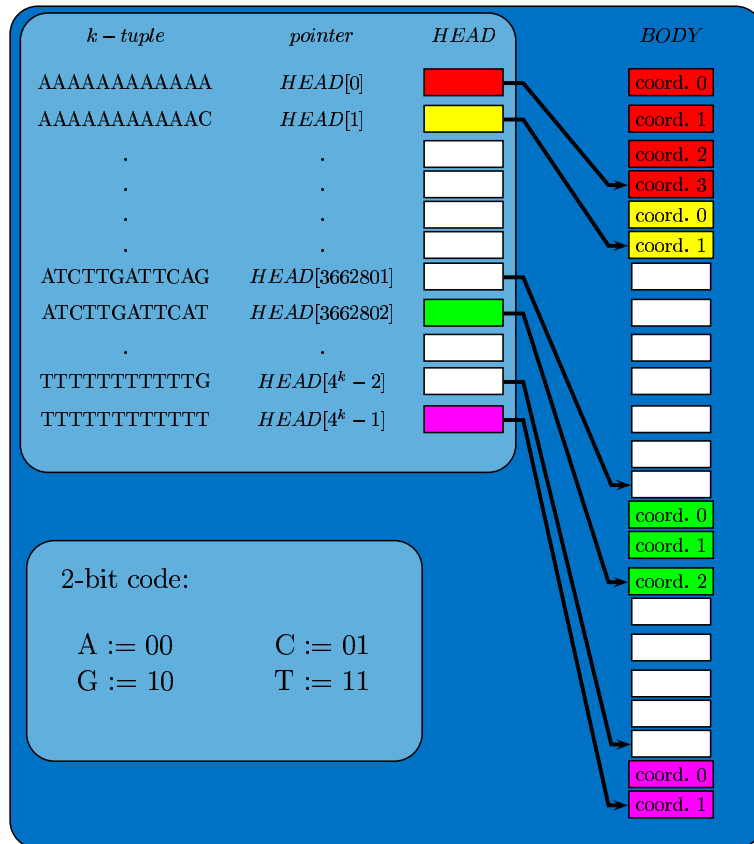




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

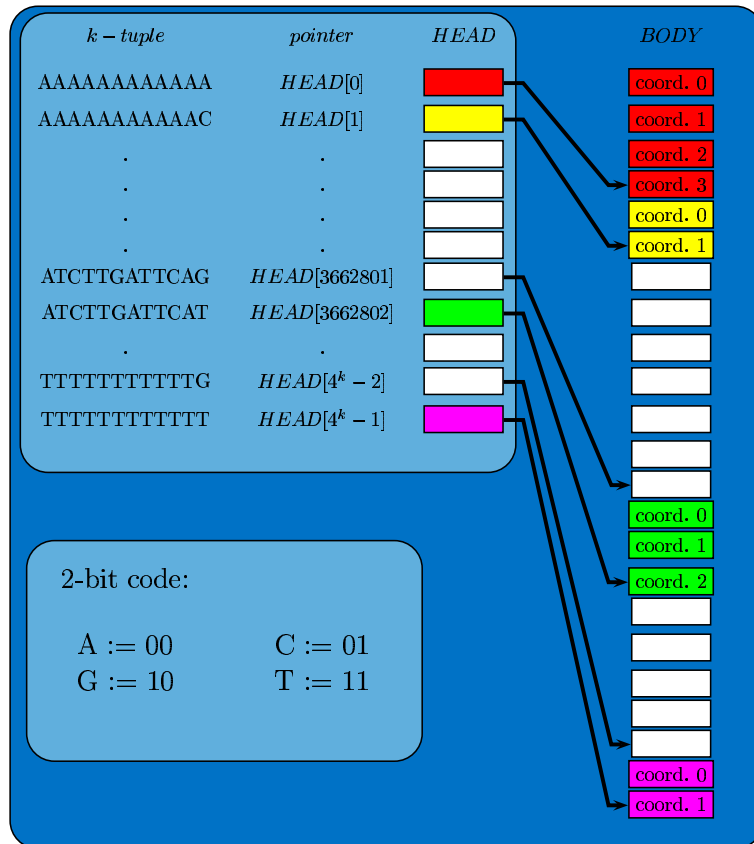
							00	11	01	11	11
--	--	--	--	--	--	--	----	----	----	----	----



- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT





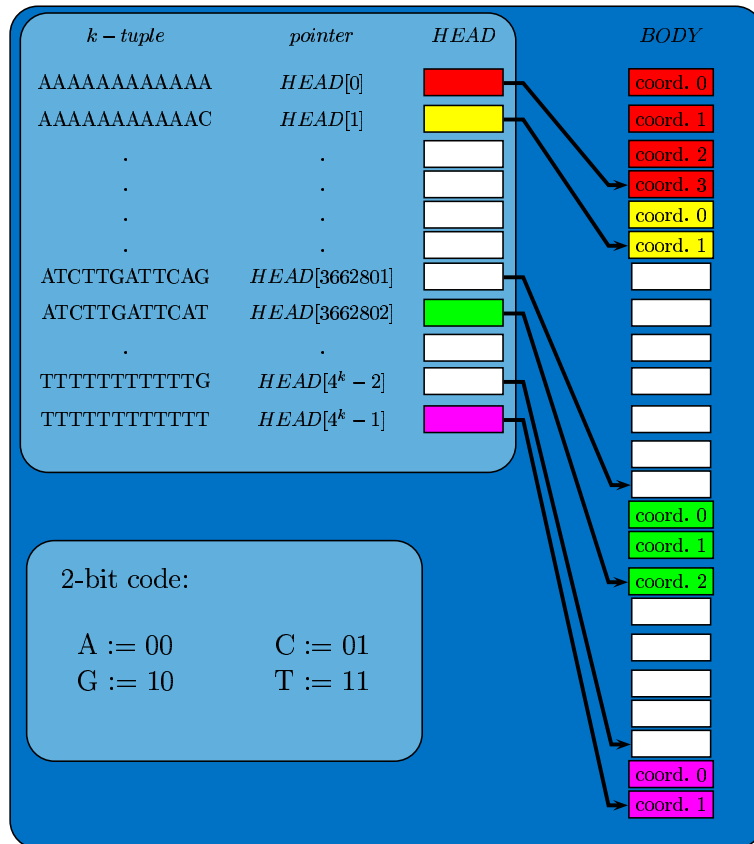
⇒ Notice that DNA has an alphabet of only 4 characters.

⇒ Can encode each base in two binary digits.

⇒ Construct an integer for each k -mer word in the sequence:

*ATCTT***G***ATTCAT*

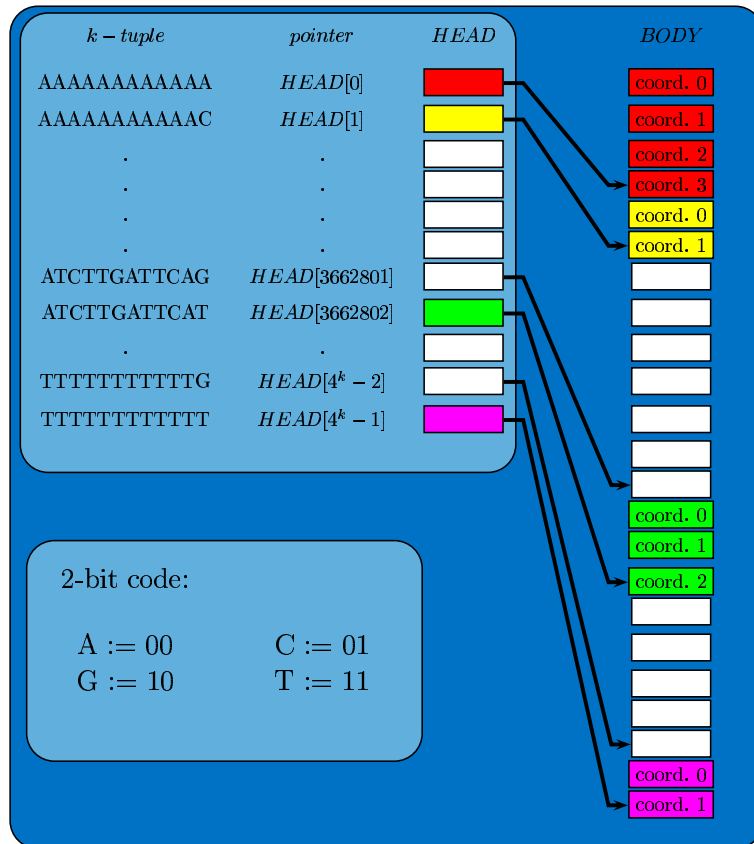
						00	11	01	11	11	10
--	--	--	--	--	--	----	----	----	----	----	----



- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

*ATCTT***G***ATTCAT*

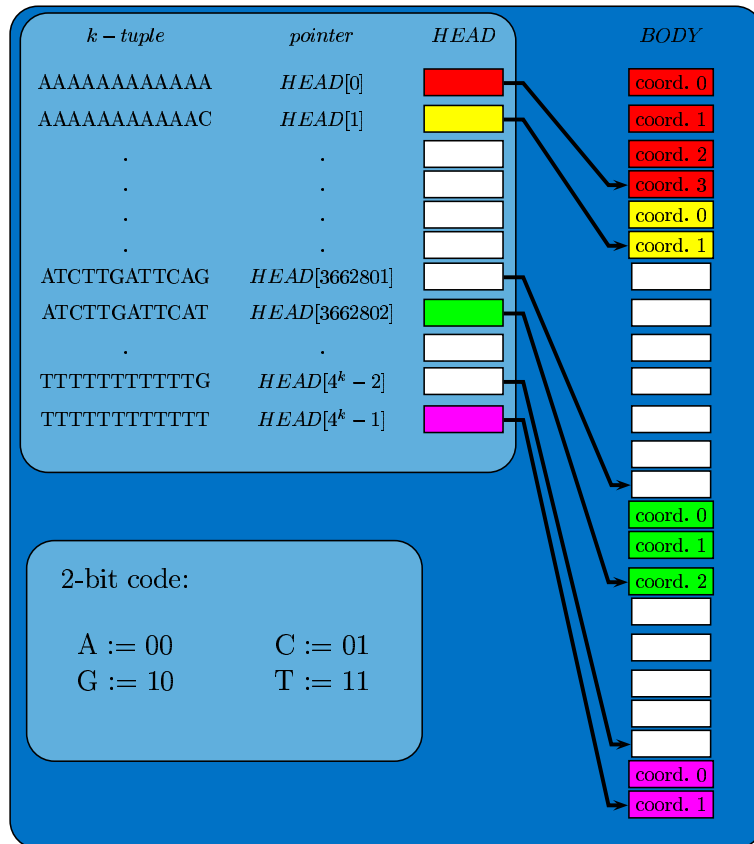




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

*ATCTTG***ATTCAT**





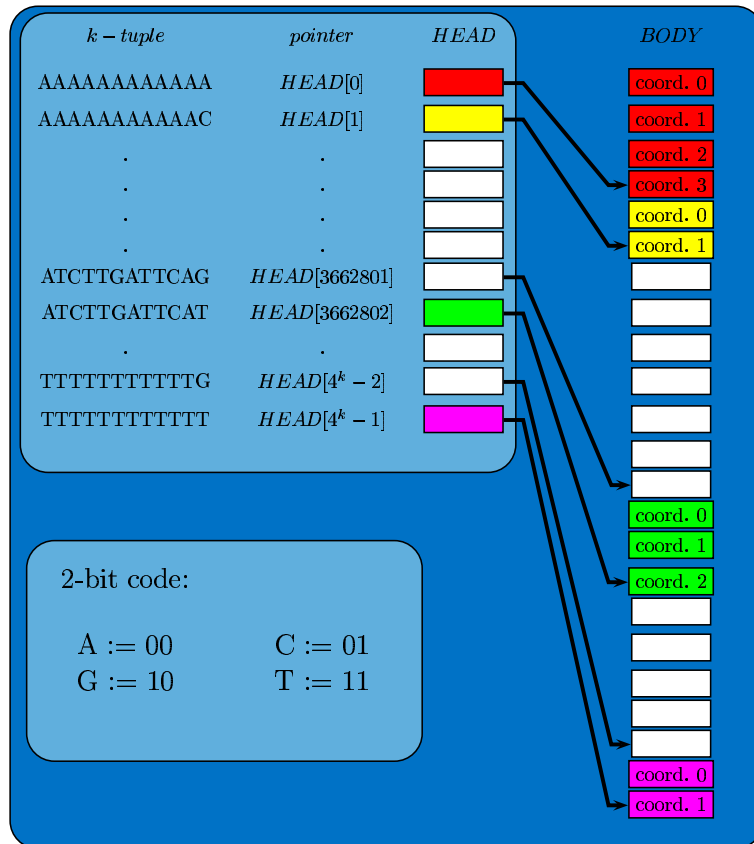
⇒ Notice that DNA has an alphabet of only 4 characters.

⇒ Can encode each base in two binary digits.

⇒ Construct an integer for each k -mer word in the sequence:

*ATCTTG***A***TTTCAT*





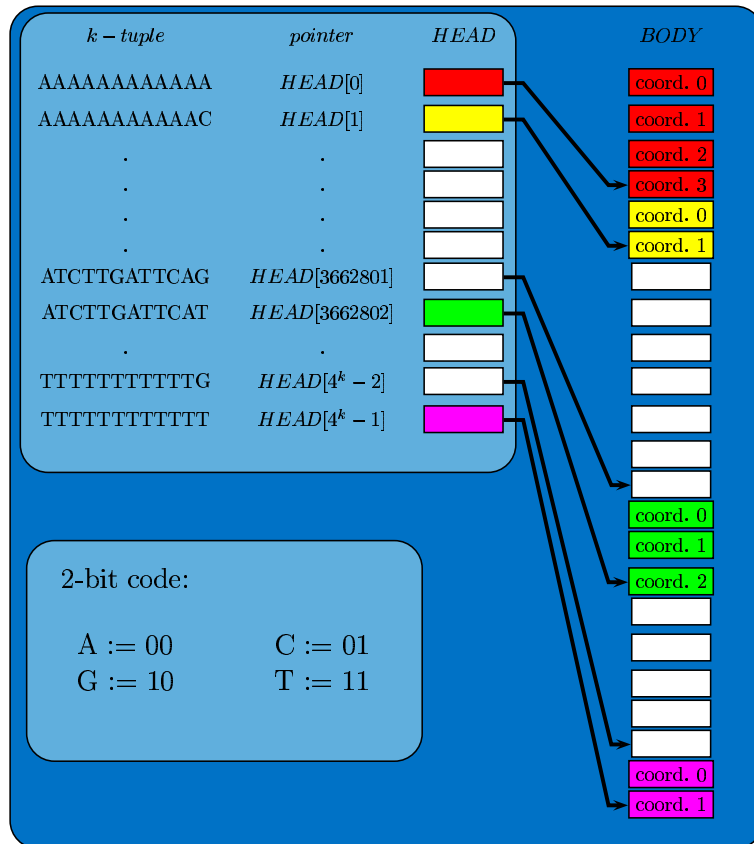
⇒ Notice that DNA has an alphabet of only 4 characters.

⇒ Can encode each base in two binary digits.

⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

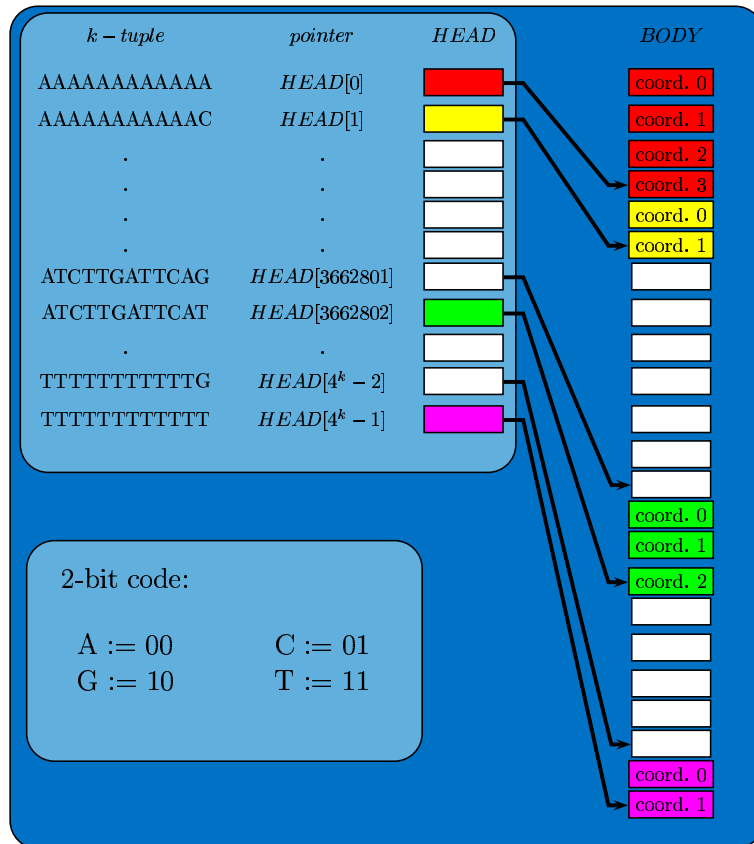




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

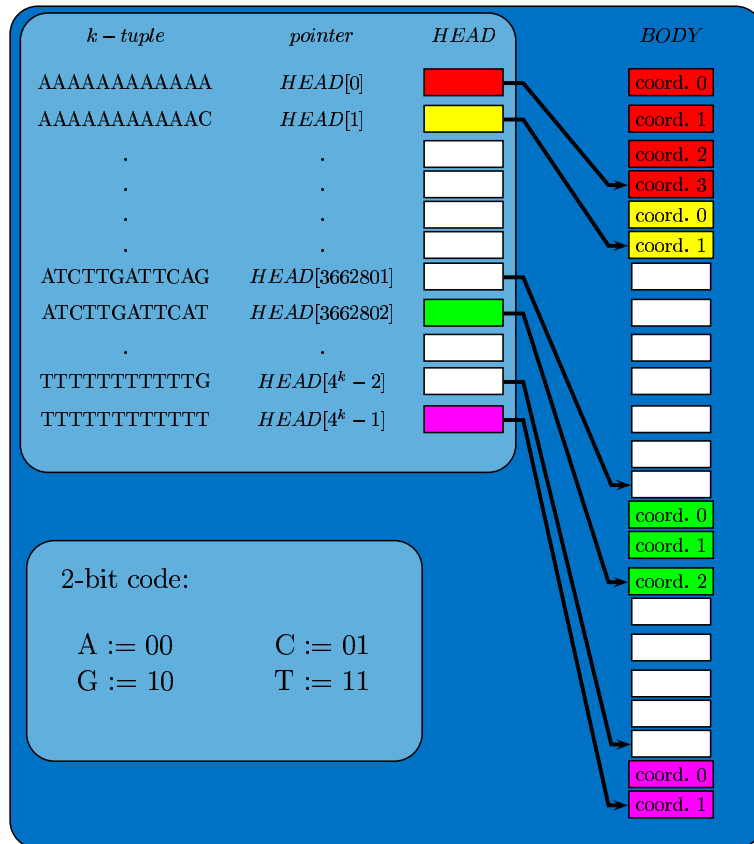




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

			00	11	01	11	11	10	00	11	11
--	--	--	----	----	----	----	----	----	----	----	----



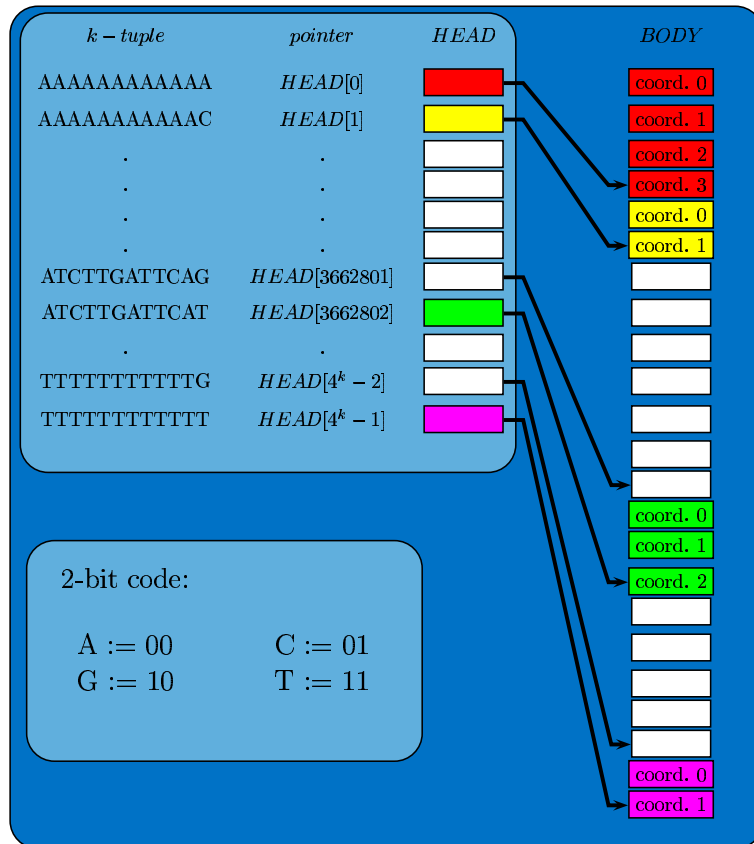
⇒ Notice that DNA has an alphabet of only 4 characters.

⇒ Can encode each base in two binary digits.

⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT





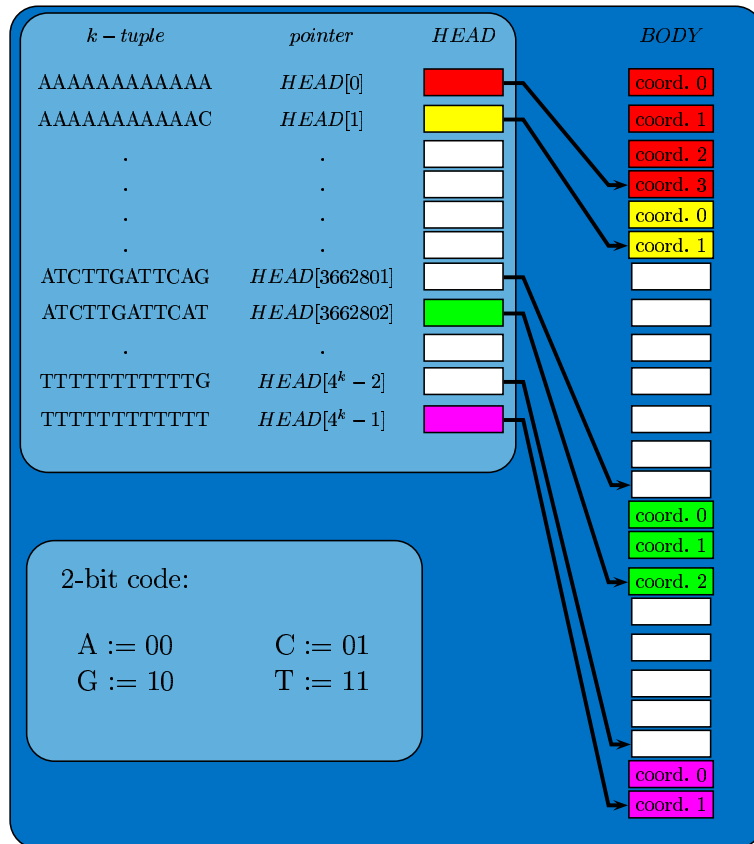
⇒ Notice that DNA has an alphabet of only 4 characters.

⇒ Can encode each base in two binary digits.

⇒ Construct an integer for each k -mer word in the sequence:

*ATCTTGATT***C***AT*

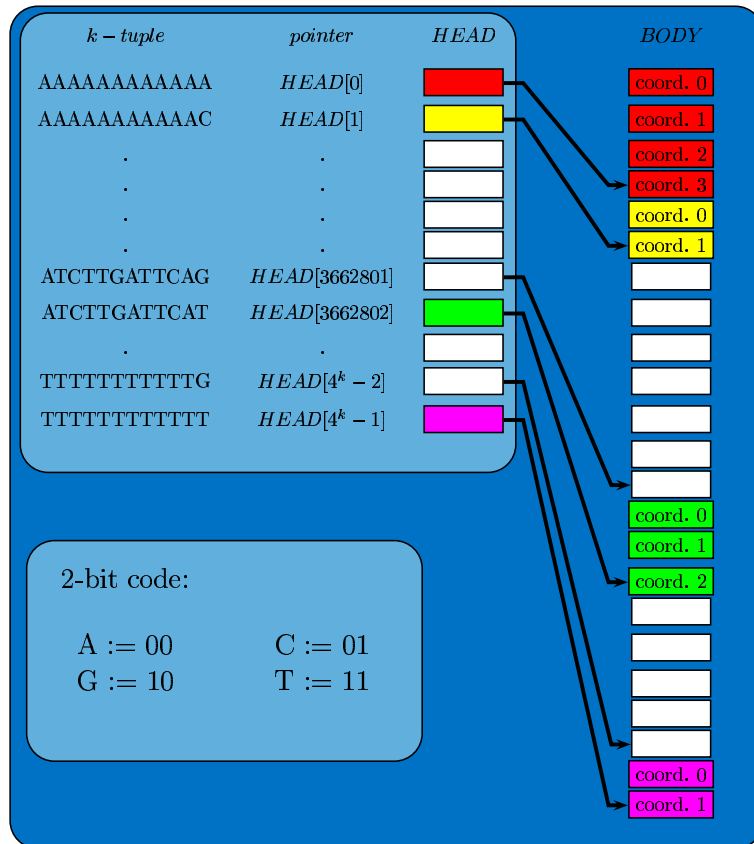
		00	11	01	11	11	10	00	11	11	01
--	--	----	----	----	----	----	----	----	----	----	----



- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

*ATCTTGATT***C***AT*

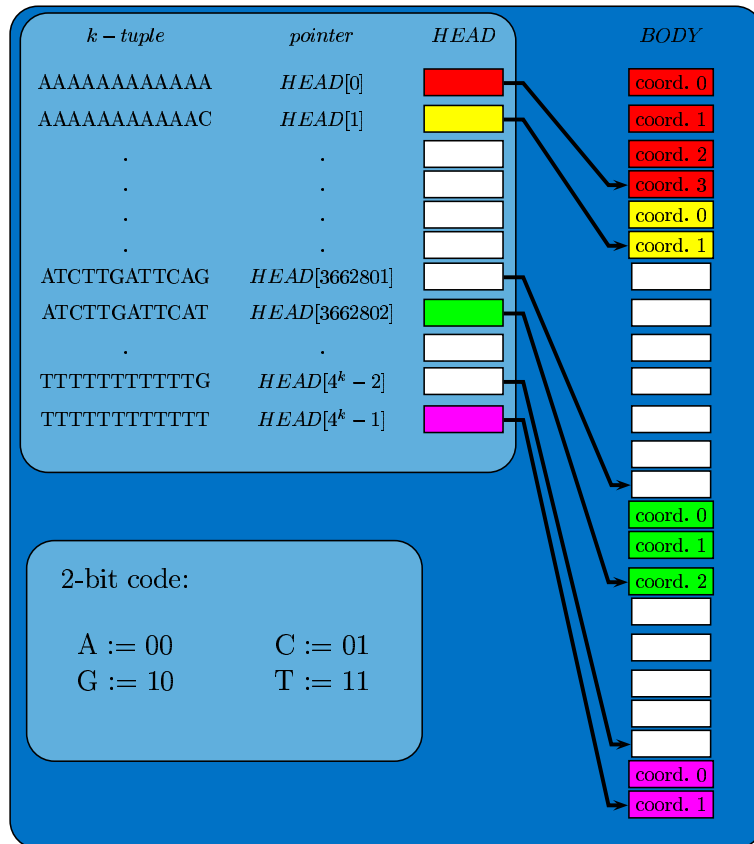




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

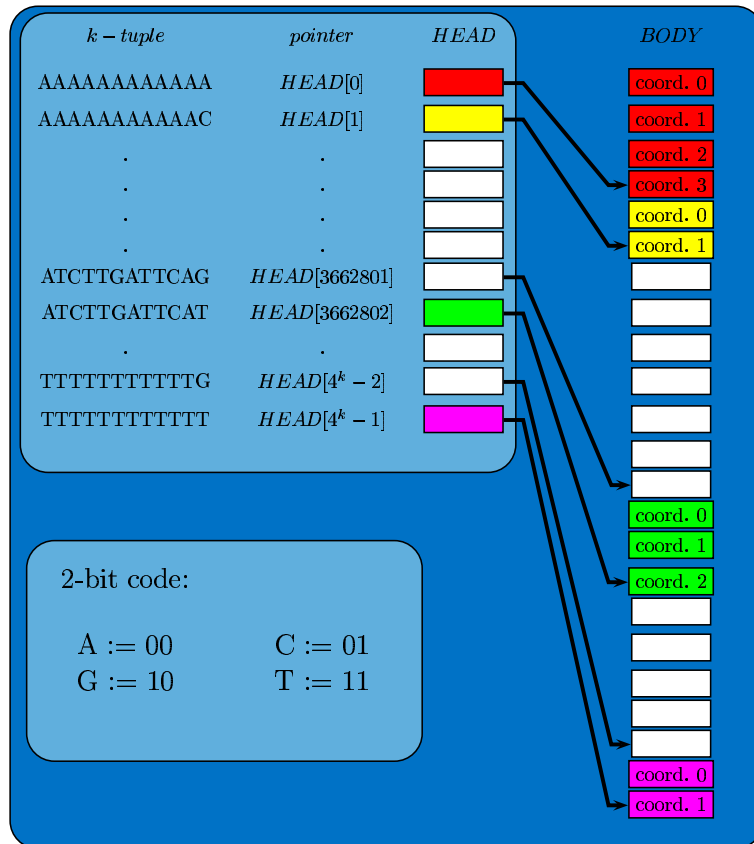




- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

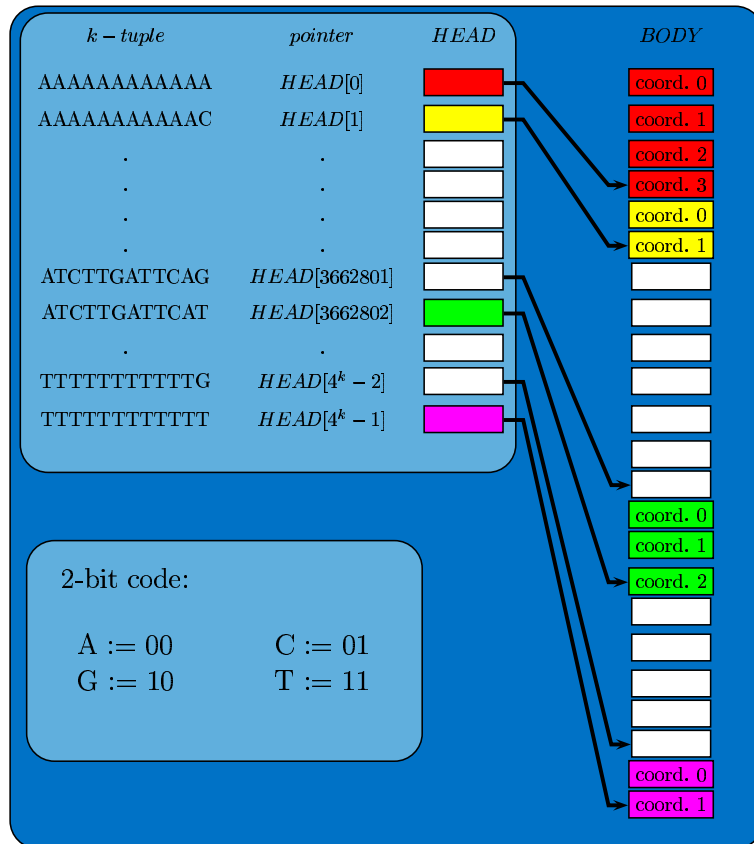
00	11	01	11	11	10	00	11	11	01	00	
----	----	----	----	----	----	----	----	----	----	----	--



- ⇒ Notice that DNA has an alphabet of only 4 characters.
- ⇒ Can encode each base in two binary digits.
- ⇒ Construct an integer for each k -mer word in the sequence:

ATCTTGATTCAT

00	11	01	11	11	10	00	11	11	01	00	11
----	----	----	----	----	----	----	----	----	----	----	----



⇒ Notice that DNA has an alphabet of only 4 characters.

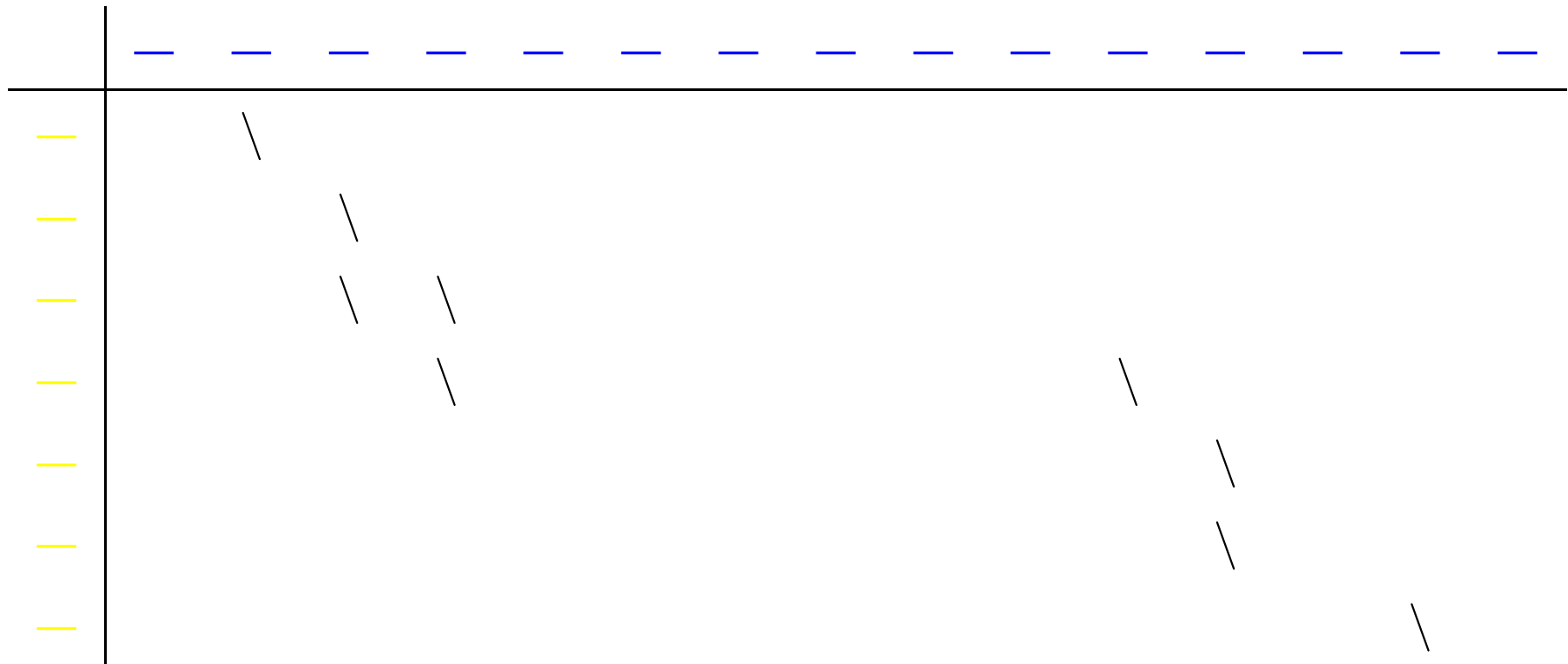
⇒ Can encode each base in two binary digits.

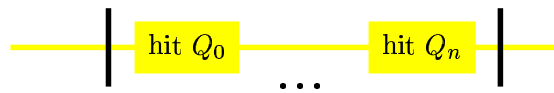
⇒ Construct an integer for each *k*-mer word in the sequence:

ATCTTGATTCAT

= 3662802



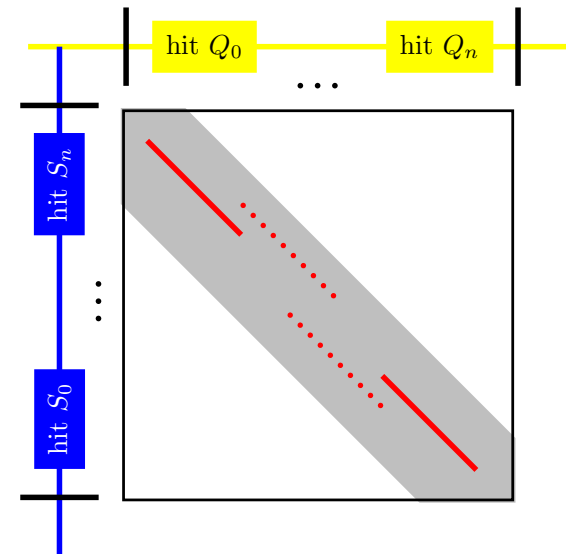
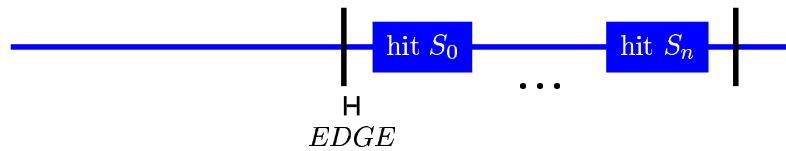




$$\text{hit } Q_i = \text{hit } S_i; \forall i$$

$$d([\text{hit } S_{i-1} - \text{hit } Q_{i-1}], [\text{hit } S_i - \text{hit } Q_i]) \leq \text{SHIFT}; i = 1, \dots, n$$

$$n \geq \text{SEEDS}$$



Download from:

<http://www.sanger.ac.uk/Software/analysis/SSAHA2/>

then:

```
$ gunzip ssaha2_1_0_1_b.mac.tar.gz
$ tar -xf ssaha2_1_0_1_b.mac.tar
$ mv ssaha2 ~/bin/
$ ssaha2 -h
```

```
deskpro272[aws]42: ./ssaha2 -h
```

```
=====
SSAHA2 1.0.1
=====
```

Copyright (C) 2003-2004 The Wellcome Trust Sanger Institute, Cambridge, UK.
All Rights Reserved.

SSAHA2 is a package combining SSAHA with cross_match developed
by Phil Green at the University of Washington.

Reference: Ning Z, Cox AJ, Mullikin JC.
SSAHA: a fast search method for large DNAdatabases.
Genome Res. 2001 Oct;11(10):1725-9.

USAGE

```
ssaha2 <query> <subject> <-kmer 12> <-seeds 5> <-skip 12> <-cut 10000>
      <-align 0> <-score 30> <-depth 50> <-memory 200>
      <-array 4> <-edge 100> <-output ssaha2> <-tags 0>
      <-sense 0> <-start 0> <-end 0> <-identity 50.0>
      <-best 0> <-stat 0> [<-save hash> <-use hash>]
```

⇒ Divide and conquer.

- ⇒ Divide and conquer.
- ⇒ Divide query sequences among available CPUs.

- ⇒ Divide and conquer.
- ⇒ Divide query sequences among available CPUs.
- ⇒ #jobs \leq #CPUs.

- ⇒ Divide and conquer.
- ⇒ Divide query sequences among available CPUs.
- ⇒ $\#jobs \leq \#CPUs$.
- ⇒ $\#jobs$ s.t. search time \leq hash time.


```
bsub -R 'select[mem>10000] rusage[mem=10000] '  
      -q hugemem -m turing -e myjob.error.%J -o myjob.out.%J  
      ./ssaha2 /not_nfs/query.fa /somewhere_considerate/chr_21.fna
```

```
sh-2.05b$ ssaha2 query.fa subject.fa \  
          > /tmp/my_sh_job.out        \  
          2> /tmp/my_sh_job.err &
```

```
[1] 373
```

```
sh-2.05b$
```

```
[mpb17391:~] aws% (ssaha2 query.fa subject.fa \  
                   > /tmp/my_csh_job.out)      \  
                   >& /tmp/my_csh_job.err &
```

```
[1] 384
```

```
[mpb17391:~] aws%
```

```
[mpb17391:~] aws% ssaha2 query.fa subject.fa -tags 1
```

```
=====
Matches For Query 0 (613 bases): ml1B-a1142a01.q1cz
=====
```

Score	Q_Name	S_Name	Q_Start	Q_End	S_Start	S_End	Direction	#Bases	identity
ALIGNMENT	604	ml1B-a1142a01.q1cz	ml1B-a1142a01.q1cz		613	1	613	F	613 99.51 613

```
Finished job
```

```
[mpb17391:~] aws% ssaha2 query.fa subject.fa \
                  -tags 1 -best 1 | \
                  grep ALIGNMENT | \
                  awk '{print $3, $2}'
```

```
ml1B-a1142a01.q1cz 604
```

```
[mpb17391:~] aws%
```

Example:

⇒ You want to align 40 million reads against the human genomes.

Example:

- ⇒ You want to align 40 million reads against the human genomes.
- ⇒ You have 12 CPUs available.

Example:

- ⇒ You want to align 40 million reads against the human genomes.
- ⇒ You have 12 CPUs available.
- ⇒ SSAHA2 will align $\sim 200,000$ reads per CPU per day.

Example:

- ⇒ You want to align 40 million reads against the human genomes.
- ⇒ You have 12 CPUs available.
- ⇒ SSAHA2 will align $\sim 200,000$ reads per CPU per day.
- ⇒ Allowing 2,400,000 per day.

Example:

- ⇒ You want to align 40 million reads against the human genomes.
- ⇒ You have 12 CPUs available.
- ⇒ SSAHA2 will align $\sim 200,000$ reads per CPU per day.
- ⇒ Allowing 2,400,000 per day.
- ⇒ Done in two weeks.

Download from:

<http://www.sanger.ac.uk/Software/analysis/SSAHA2/>

then:

```
$ gunzip ssaha2Srv_1_0_1_c.mac.tar.gz
$ tar -xf ssaha2Srv_1_0_1_c.mac.tar.gz
$ mv ssaha* ~/bin/
$ ssaha2Server -h
```

Ensembl Genome Browser

http://www.ensembl.org/index.html

journals info webmail travel linux mac programming etc ssaha tools maths bioinformatics office news conferences

Search all Ensembl: Anything **Go**

e! Ensembl

Ensembl v37 - Feb 2006 **Help**

Use Ensembl to...

- Run a BLAST search
- Search Ensembl
- Data mining [BioMart]
- Upload your own data
- Export data
- Download data

Docs and downloads

- Information
- What's New
- About Ensembl
- Ensembl data
- Software

Other links

- Home
- Sitemap
- Vega
- Pre Ensembl
- View previous release of page in Archive!
- Stable Archive! link for this page
- Archive! sites
- Trace server

What's New in Ensembl 37

- New mosquito assembly and genebuild (*Anopheles gambiae*)
- New Xenopus assembly and genebuild (*Xenopus tropicalis*)
- New Ciona assembly and genebuild (*Ciona intestinalis*)
- TranscriptSNPView (*Mus musculus*)
- GeneSealView (all species)

[More news...](#)

About Ensembl

Ensembl is a joint project between [EMBL - EBI](#) and the [Sanger Institute](#) to develop a software system which produces and maintains automatic annotation on selected eukaryotic genomes. Ensembl is primarily funded by the [Wellcome Trust](#).

This site provides [free access](#) to all the data and software from the Ensembl project. Click on a species name to browse the data.

Access to all the data produced by the project, and to the software used to analyse and present it, is provided free and without constraints. Some data and software may be subject to [third-party constraints](#).

For all enquiries, please [contact the Ensembl HelpDesk](#) (helpdesk@ensembl.org).

Other sites using the Ensembl system

- [EBI Genome Reviews](#) database - mainly archaea and bacteria.
- [VEGA](#) - Vertebrate Genome Annotation

[More...](#)

Mammalian genomes

- Homo sapiens* NCBI 35 | Vega | **pre!**
- Pan troglodytes* PanTro 1.0
- Macaca mulatta* MMUL 0.1 | **pre!**
- Mus musculus* NCBI m34 | Vega | **pre!**
- Rattus norvegicus* RGC 3.4
- Pre!** *Oryctolagus cuniculus* **NEW!** RABBIT
- Canis familiaris* CanFam 1.0 | Vega | **pre!**
- Bos taurus* Btau 2.0
- Pre!** *Dasypus novemcinctus* **NEW!** ARM1A
- Pre!** *Loxodonta africana* BROAD E1
- Pre!** *Echinops telfairi* **NEW!** TENREC
- Monodelphis domestica* MonDom 2.0

Other species

- Gallus gallus* WASHUC 1
- Xenopus tropicalis* **UPDATED!** JGI 4
- Danio rerio* Zv5 | Vega | **pre!**
- Fugu rubripes* FUGU 4.0
- Tetraodon nigroviridis* TETRAODON 7
- Ciona intestinalis* **UPDATED!** JGI2
- Pre!** *Ciona savignyi* CSAV 2.0
- Drosophila melanogaster* BDGP 4
- Anopheles gambiae* **UPDATED!** AgamP3
- Pre!** *Aedes aegypti* AEDES 1
- Apis mellifera* Amel 2.0
- Caenorhabditis elegans* **UPDATED!** WS 150
- Saccharomyces cerevisiae* SGD 1

Echinops telfairi

Done

© 2006 WTSI / EBI. Ensembl is available to [download for public use](#) - please see the [code licence](#) for details.

BlastView

http://www.ensembl.org/Multi/blastview

journals info webmail travel linux mac programming etc ssaha tools maths bioinformatics office news conferences

Search all Ensembl: Anything Go

e! Ensembl Multi BlastView

Ensembl v37 - Feb 2006

Use Ensembl to...

- Run a BLAST search
- Search Ensembl
- Data mining [BioMart]
- Upload your own data
- Export data
- Download data

Docs and downloads

- Information
- What's New
- About Ensembl
- Ensembl data
- Software

Other links

- Home
- Sitemap
- Vega
- Pre Ensembl
- View previous release of page in Archive!
- Stable Archive! link for this page
- Archive! sites
- Trace server

Summary

- setup (Not yet initialised)
- configure (Not yet initialised)
- results (Not yet initialised)
- display (Not yet initialised)

new SETUP CONFIG RESULTS DISPLAY

refresh Online Help

Enter the Query Sequence

Either Paste sequences (max 30) in FASTA or plain text:

Or Upload a file containing one or more FASTA sequences

Browse...

Or Enter a sequence ID or accession (EMBL, UniProt, RefSeq)

Retrieve

Or Enter an existing ticket ID:

Retrieve

dna queries
peptide queries

Select the databases to search against

Select species:
Use 'ctrl' key to select multiple species

Anopheles_gambiae
Apis_mellifera
Bos_taurus

dna database: Genomic sequence
peptide database: Ensembl Peptides

Select the Search Tool

BLASTN
SSAHA2
TBLASTX

configure RUN

Search sensitivity:
Optimise search parameters to find the following alignments

Near-exact matches

About BlastView

BlastView provides an integrated platform for sequence similarity searches against Ensembl databases, offering access to both BLAST and SSAHA programs. [More](#)

We would like to hear your impressions of BlastView, especially regarding functionality that you would like to

Done

The screenshot displays the Ensembl Multi BlastView web interface. The browser address bar shows the URL http://www.ensembl.org/Multi/blastview/BLA_DKUP1286e. The page title is "Ensembl v37 - Feb 2006".

Left Sidebar:

- Use Ensembl to...**
 - Run a BLAST search
 - Search Ensembl
 - Data mining [BioMart]
 - Upload your own data
 - Export data
 - Download data
- Docs and downloads**
 - Information
 - What's New
 - About Ensembl
 - Ensembl data
 - Software
- Other links**
 - Home
 - Sitemap
 - Vega
 - Pre Ensembl
 - View previous release of page in Archive!
 - Stable Archive! link for this page
 - Archive! sites
 - Trace server

Top Navigation: new | SETUP | CONFIG | RESULTS | DISPLAY

Main Content Area:

Displaying *ml18-a1142a01.q1cz* sequence alignments vs *Homo_sapiens* LATESTGP database
Showing top 100 alignments of 38, sorted by Raw Score

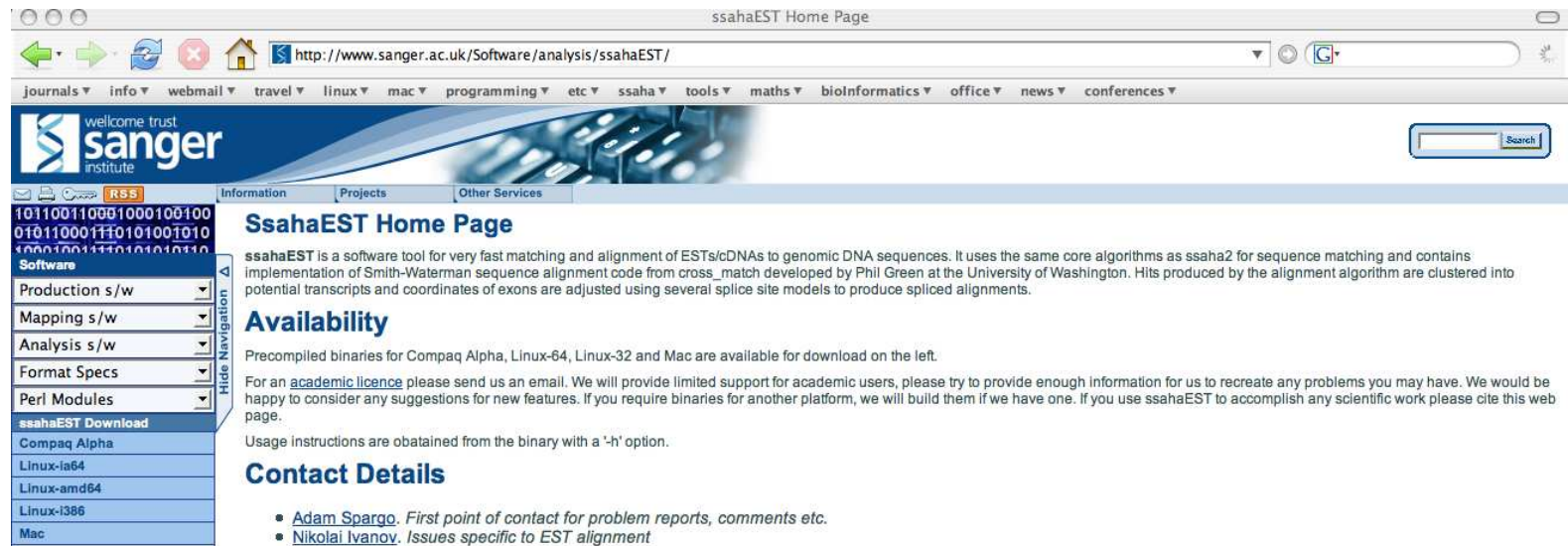
☒ Alignment Locations vs. Karyotype (click arrow to hide)

☒ Alignment Locations vs. Query (click arrow to hide)

Right Sidebar:

- Summary**
 - setup**
 - Homo_sapiens
 - Genomic sequence
 - SSAHA2
 - Low sensitivity
 - configure**
 - depth: 100
 - seeds: 2
 - score: 20
 - results**
 - display**
 - Not yet initialised

Bottom Left: Ciona intestinalis 2.0



```
deskpro272[aws]68: ls
```

```
chr_10_89612000_89722000.fna
```

```
MTX
```

```
HUMAN_PTEN.fna
```

```
README
```

```
HUMAN_PTEN.fna_chr_10_89612000_89722000.fna.ssahaEST ssahaEST
```

```
deskpro272[aws]69: ./ssahaEST HUMAN_PTEN.fna chr_10_89612000_89722000.fna
```




=====
Matches For Query 0 (592 bases): 20SNP45079-1010a03.p1c
=====

Score	Q_Name	S_Name	Q_Start	Q_End	S_Start	S_End	Direction	#Bases	identity
514	20SNP45079-1010a03.p1c	20.1-62435964	32	592	28177108	28177668	F	561	98.40 592

ProcessSNP_start 20SNP45079-1010a03.p1c

snp_start 20.1-62435964_28177358

ssaha:SNP 20.1-62435964 0 20SNP45079-1010a03.p1c A G 40 40 28177357 280 1 28177108 28177668 0 62435964

alignment name

alignment 20SNP45079-1010a03.p1c AACAAACCCAAATGTTACCCAGTAGATGAACGGATAAACAAA

alignment 20.1-62435964 AACAAACCCAAATGTTACCAATAGATGAACGGATAAACAAA

snp_end 20SNP45079-1010a03.p1c_28177358

ProcessSNP_end 20SNP45079-1010a03.p1c

ProcessIndel_start 20SNP45079-1010a03.p1c 0 5

ssaha:indel 20SNP45079-1010a03.p1c 20.1-62435964 0 28177608 500 1 1 N - 28177108 28177668 0 62435964 40 A 40 N 40 G 40 G

alignment name

alignment 20SNP45079-1010a03.p1c AGAAGGAATATGGGGGGATAGGGGAGGTGATANCTAAAAGT

alignment 20.1-62435964 AGAAGGAATATGGGGGGATA-GGGAGGTGATAGCTAAAAGT

ProcessIndel_end 20SNP45079-1010a03.p1c

Wellcome Trust Sanger Institute Press Releases

http://www.sanger.ac.uk/Info/Press/2006/060321.shtml

journals info webmail travel linux mac programming etc ssaha tools maths bioinformatics office news conferences

wellcome trust sanger institute

Information Projects Other Services

Press Releases: 21st March 2006

TraceSearch - Google for the genome Sanger Institute's 100-fold faster DNA search engine

The Wellcome Trust Sanger Institute launches today a search engine that speeds up the hunt for important DNA sequences more than 100-fold. Just as important, the engine scans all known sequence data from all available organisms, saving time for researchers, who, until the release of TraceSearch, had to inspect each organism in turn.

When studying genes, biomedical researchers often need to find related sequences in the public databases because those might hold clues to the function of the gene, or might identify a critical variant involved in disease. If the sequences are not complete, which is true for most large genomes, the key to finding the parts not in the assembled sequence is to search the original data - the sequence 'traces'. The new engine makes that comparison much more efficient (<http://trace.ensembl.org/cgi-bin/tracesearch>).

Until the development of the new search system, the only option was to query sequences from one species at a time: those queries were placed in a queue and the results returned after 10 minutes or longer. The new search system searches all species and works 'live', returning results in just 3-5 seconds.

"The TraceSearch achieves its speed by indexing the DNA sequences, rather than performing a computationally expensive alignment for all sequences in the database: we only ever do that for the very best hits from the index," explained Dr Adam Spargo, senior developer of TraceSearch. "Classical methods of DNA alignment rely on dynamic programming methods which slow down dramatically as the size of the database increases. Even conventional indexing methods produce a massive index which has to be accessed from a hard disk."

"We use the SSAHA algorithm, which exploits the fact that DNA has an alphabet of only four letters, to construct a small index which fits into the RAM of the computer. As the database grows the time to do an index look-up remains short and we only ever perform a few alignments."

Dr Zemin Ning "This is an exciting project. The new search engine is not just an improvement, but a dramatic reinvention of the earlier engine."

However, even using the SSAHA method the index is large - currently 300 Gigabytes - and the team made the decision to distribute it over a cluster of Linux machines, offering an inherently scalable and economical solution. The hashtable is hosted on 30 IBM LS20 blades.

This marks a transition from dynamic programming-based algorithms that run on a single large machine to a distributed index approach that runs on a cluster of commodity machines, with the final alignment phase being akin to the PageRank phase in Google, ordering the hits depending on relevance to query.

"The system not only finds absolute matches, but finds related matches," continued Dr Spargo. "Variation in DNA sequence is vital in disease studies and we needed to build in the ability to find all sequences related to the query sequence."

This remarkable performance is built on the Sanger Institute Trace Archive, a repository of almost one trillion DNA bases. Printed on A4 paper, there would be 133 million sheets, covering the City of London three times; a single stack of paper would be 2.5 times the height of Mount Everest. TraceSearch, however, would find any piece of DNA text in a few seconds. The Sanger Institute Trace Archive collaborates with the NCBI Trace Archive to act as a global repository for all unassembled DNA sequence data. The Trace Archives complement the International Sequence Databases EMBL/Genbank/DBJ which are the repository for assembled and annotated sequence records.

"Speed and cross-species search are the two flashing points for TraceSearch," explained Dr Zemin Ning, the project leader who is responsible for the development team at the Institute. "We had previously built SSAHA single hashtables for trace search before, but we knew we would need dramatic improvements if we were to cope with growing sequence data and demands of researchers for rapid access to sequence from all species."

"This is an exciting project. The new search engine is not just an improvement, but a dramatic reinvention of the earlier engine. The alignment quality has been significantly improved without a slowdown in match speed, given the size of the database. Since no genome has been 100% finished, TraceSearch users can find holes in the genome and cross-species search may play also a role in comparative genomics."

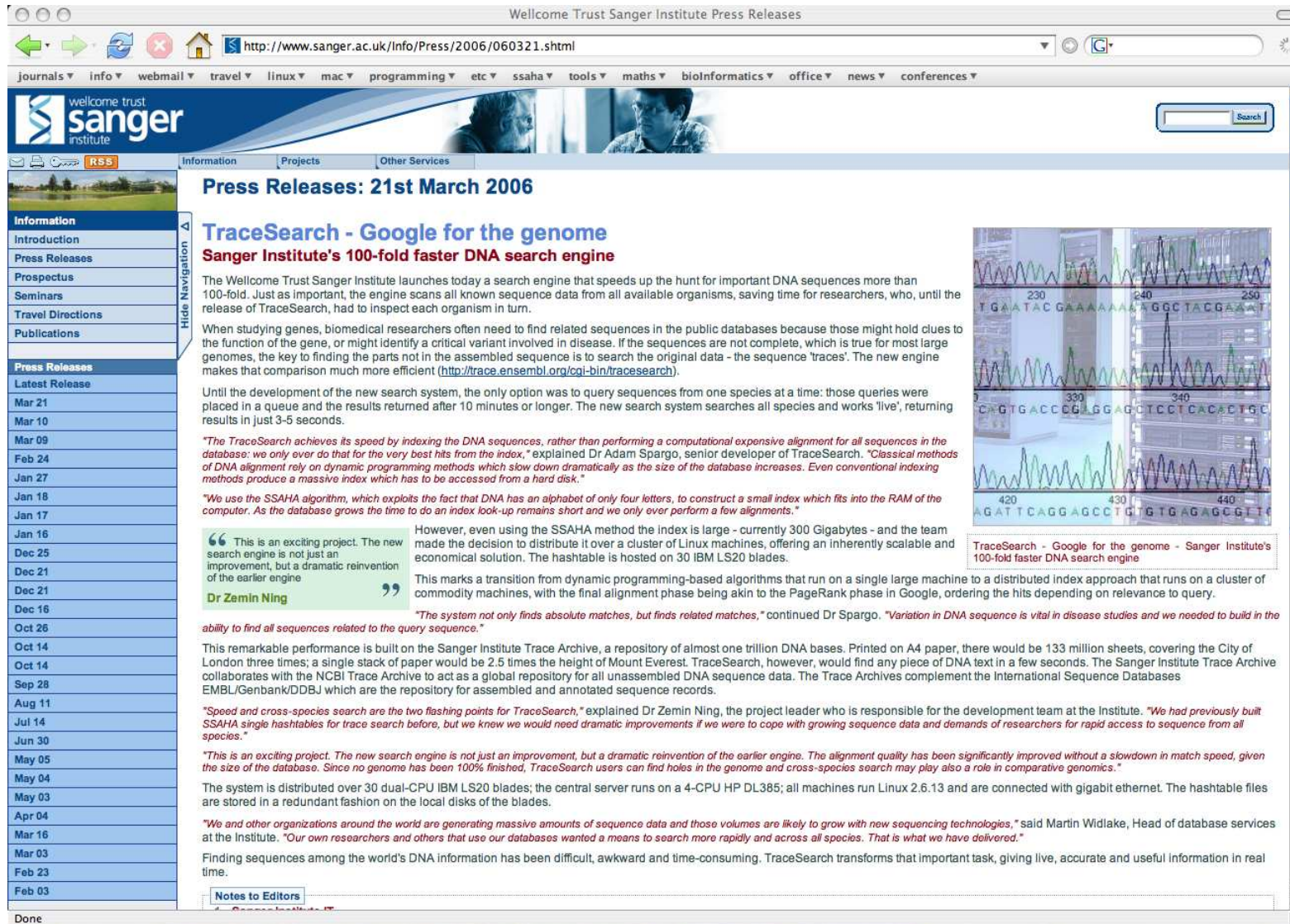
The system is distributed over 30 dual-CPU IBM LS20 blades; the central server runs on a 4-CPU HP DL385; all machines run Linux 2.6.13 and are connected with gigabit ethernet. The hashtable files are stored in a redundant fashion on the local disks of the blades.

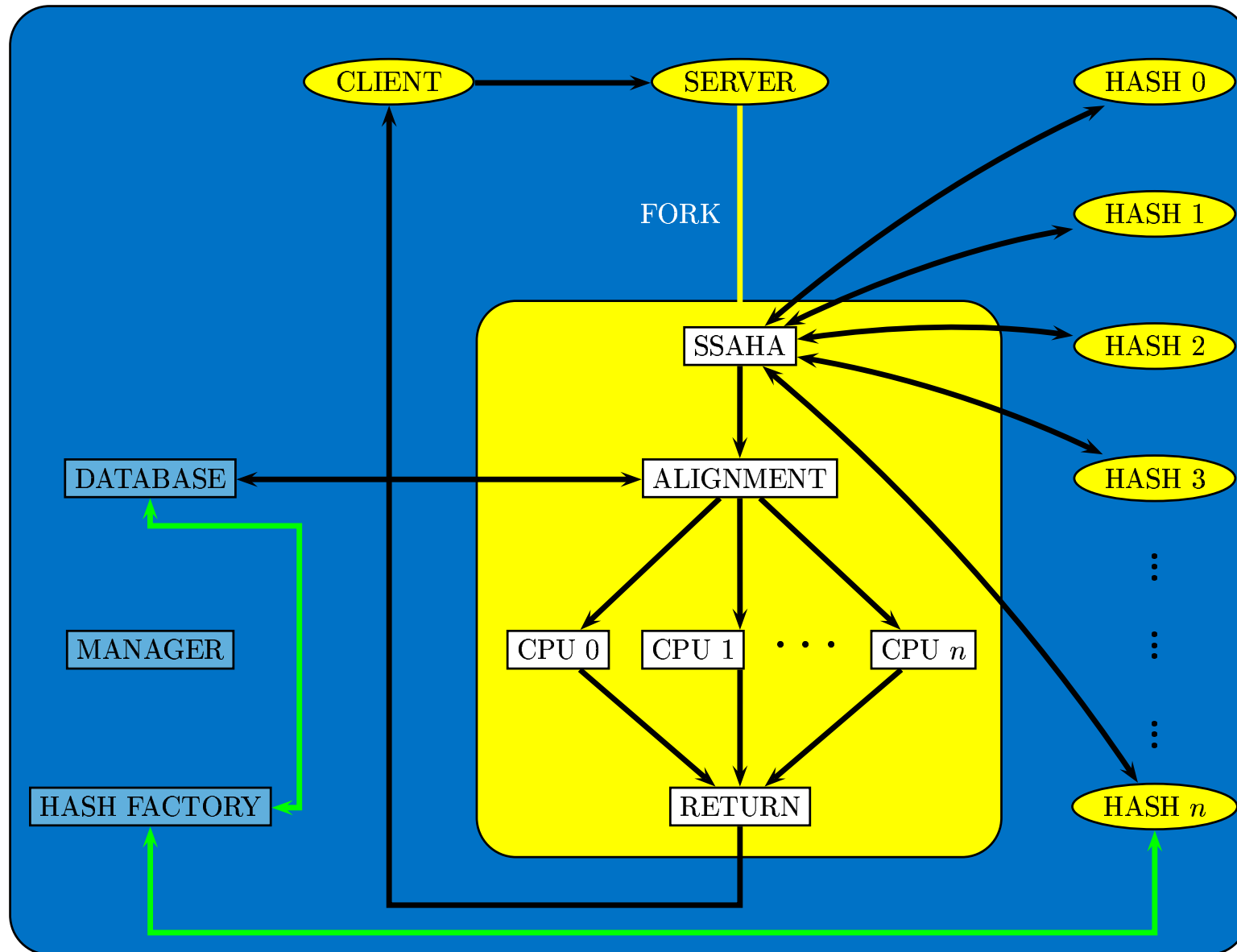
"We and other organizations around the world are generating massive amounts of sequence data and those volumes are likely to grow with new sequencing technologies," said Martin Widelake, Head of database services at the Institute. "Our own researchers and others that use our databases wanted a means to search more rapidly and across all species. That is what we have delivered."

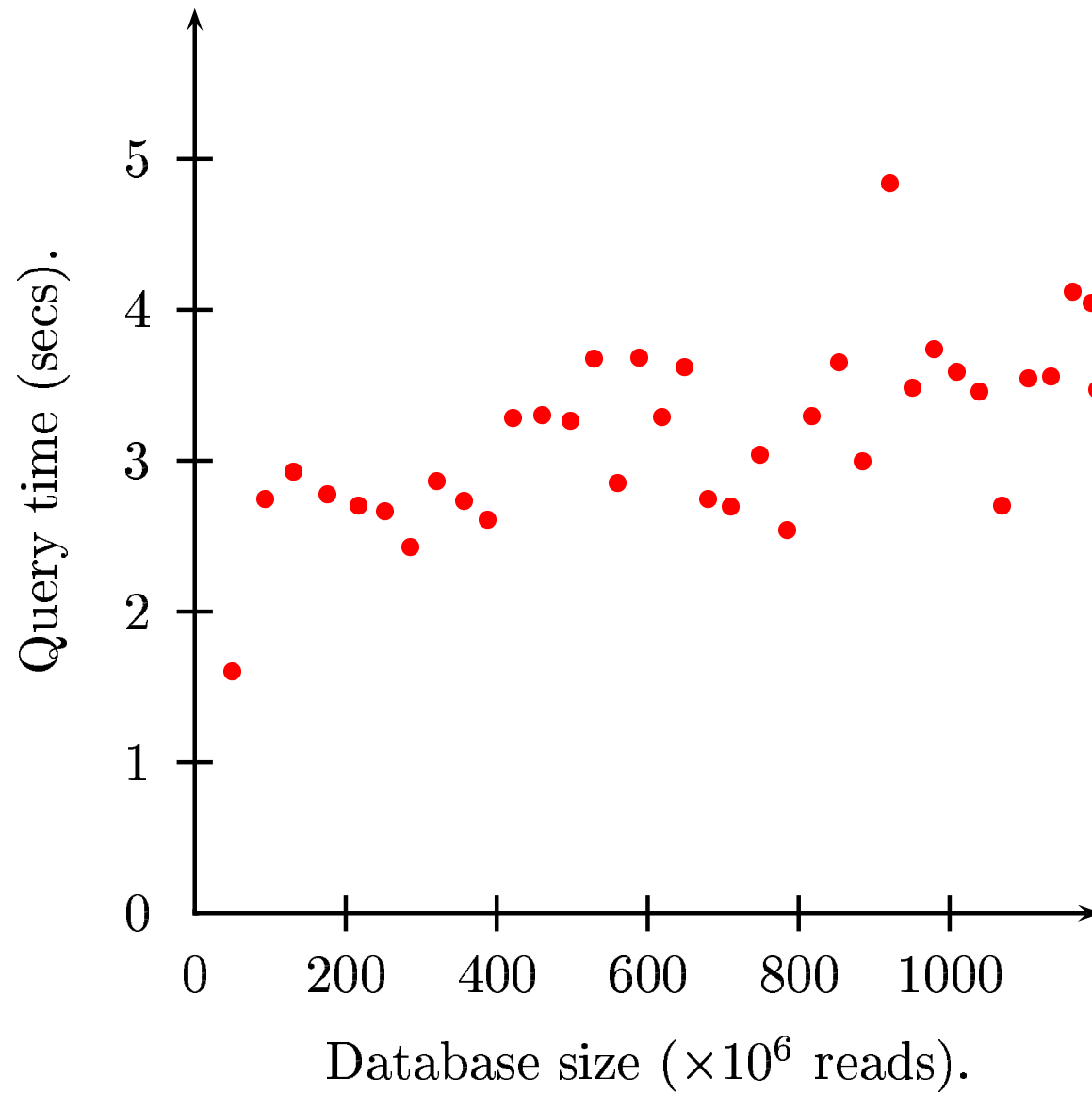
Finding sequences among the world's DNA information has been difficult, awkward and time-consuming. TraceSearch transforms that important task, giving live, accurate and useful information in real time.

Notes to Editors

Done







Trace Search

http://trace.ensembl.org/cgi-bin/tracesearch

journals info webmail travel linux mac programming etc ssaha tools maths bioinformatics office news conf

e!Ensembl Trace Server

Trace Search

- Home page
- View trace statistics
- Search trace sequences
- Download traces (FTP)
- FAQ

Links

- Ensembl
- NCBI trace archive

Existing ticket id: **retrieve** **New Search**

FASTA Query Sequence(s)

paste a sequence below:

or upload a file **Browse...**

Options

species	any		
centre	any	output	ssaha2
type	any	sort	score
seeds	5	depth	50

show alignments ☐

search

Trace Search

http://trace.ensembl.org/cgi-bin/tracesearch?id=37343

journals info webmail travel linux mac programming etc ssaha tools maths bioinformatics office news conferences

e!Ensembl Trace Server


Find trace: **Go**
e.g. [m1B-a1798c05.glc](#), [GTMZP10038*](#)

Trace Search

- Home page
- View trace statistics
- Search trace sequences
- Download traces (FTP)
- FAQ

Links

- Ensembl
- NCBI trace archive



Existing ticket id: 37343 **retrieve** **New Search**

- Search Status

Status	DONE	Position in Queue	0
Options	-depth=50 -seeds=5 -output=ssaha2	Pending jobs	0
	-sort=score	Running jobs	0
Submitted	6th Apr 2006 1147	Avg. throughput (jobs/sec)	0.00
Started	6th Apr 2006 1147	Wallclock queued wait time	00:00:00
Ended	6th Apr 2006 1147	Wallclock execution time	00:00:09
		Wallclock total time	00:00:09

+ Query Sequence(s)

- Search Results

Matches For Query 0 (613 bases): m1B-a142a01.glc

Score	Q_Name	S_Name	Q_Start	Q_End	S_Start	S_End	Direction	#Bases	identity
604	m1B-a142a01.glc	m1B-a142a01.glc	1	613	1	613	F	613	99.51 613
492	m1B-a142a01.glc	m1B-a142a01.glc	1	583	2	589	F	583	95.20 613
477	m1B-a142a01.glc	G10P678365RD1.T0	41	586	133	680	F	546	96.15 613
477	m1B-a142a01.glc	ojz19a02.bl	41	586	201	748	F	546	96.15 613
477	m1B-a142a01.glc	L25824P6011RP6.T0	41	586	697	150	C	546	96.15 613
477	m1B-a142a01.glc	jeF73a09.gl	41	586	82	629	F	546	96.15 613
474	m1B-a142a01.glc	L9866816084236	41	586	110	657	F	546	95.97 613
470	m1B-a142a01.glc	L25824P6010RC4.T0	41	586	200	747	F	546	95.60 613
467	m1B-a142a01.glc	L25824P6008RA9.T0	41	586	698	153	C	546	95.79 613
462	m1B-a142a01.glc	L9866848182006	41	586	744	194	C	546	95.42 613
459	m1B-a142a01.glc	lmo12a03.bl	41	586	680	136	C	546	95.42 613
451	m1B-a142a01.glc	G10P62512SRAB.T0	41	586	250	718	F	468	99.15 613
449	m1B-a142a01.glc	L25824P6007RG11.T0	41	586	1138	588	C	546	94.51 613
448	m1B-a142a01.glc	krv03h12.gl	41	558	250	767	F	518	95.95 613
443	m1B-a142a01.glc	mse39371-828b06.glc	58	586	72	600	F	529	94.90 613
441	m1B-a142a01.glc	L25824P6007RG9.T0	41	586	849	302	C	546	94.14 613
425	m1B-a142a01.glc	L25824P6002RF4.T0	41	525	240	725	F	485	96.08 613
422	m1B-a142a01.glc	ind86c01.bl	41	586	113	660	F	546	92.67 613
418	m1B-a142a01.glc	L9866846414296	101	586	1	487	F	486	95.88 613
390	m1B-a142a01.glc	ojz10a08.gl	41	437	488	91	C	397	99.50 613
381	m1B-a142a01.glc	L25824P6004RB10.T0	41	438	431	32	C	398	98.74 613
381	m1B-a142a01.glc	G10P642886RH4.T0	142	586	752	307	C	445	95.73 613
376	m1B-a142a01.glc	L25824P6003RB10.T0	41	431	429	39	C	391	96.98 613
372	m1B-a142a01.glc	ojz07a02.bl	151	586	47	483	F	436	95.64 613
372	m1B-a142a01.glc	itm42h09.gl	125	586	1025	565	C	462	93.94 613
370	m1B-a142a01.glc	ojz19d12.bl	41	532	446	927	F	492	93.50 613
369	m1B-a142a01.glc	G10P670226FP4.T0	41	426	336	722	F	386	98.45 613

[illegible]

Trace Search

http://trace.ensembl.org/cgi-bin/tracesearch?id=37343

journals info webmail travel linux mac programming etc ssaha tools maths bioinformatics office news conferences

Matches For Query 0 (613 bases): m11B-all142a01.q1oz

Score	Q_Name	S_Name	Q_Start	Q_End	S_Start	S_End	Direction	#Bases	identity
604	m11B-all142a01.q1oz	m11B-all142a01.q1oz	1	613	1	613	F	613	99.51 613
492	m11B-all142a01.q1oz	m11B-all142a01.q1oz	1	583	2	589	F	583	95.20 613
477	m11B-all142a01.q1oz	G10P678365RD1.T0	41	586	133	680	F	546	96.15 613
477	m11B-all142a01.q1oz	o1z19a02.b1	41	586	201	748	F	546	96.15 613
477	m11B-all142a01.q1oz	L25824P6011RP6.T0	41	586	697	150	C	546	96.15 613
477	m11B-all142a01.q1oz	1a7f3a09.q1	41	586	82	629	F	546	96.15 613
474	m11B-all142a01.q1oz	19866816084236	41	586	110	657	F	546	95.97 613
470	m11B-all142a01.q1oz	L25824P6010RC4.T0	41	586	200	747	F	546	95.60 613
467	m11B-all142a01.q1oz	L25824P6008RA9.T0	41	586	698	153	C	546	95.79 613
462	m11B-all142a01.q1oz	19866848182006	41	586	744	194	C	546	95.42 613
459	m11B-all142a01.q1oz	1mo12a03.b1	41	586	680	136	C	546	95.42 613
451	m11B-all142a01.q1oz	G10P6251288A8.T0	41	586	250	718	F	468	99.15 613
449	m11B-all142a01.q1oz	L25824P6007RG11.T0	41	586	1138	588	C	546	94.51 613
448	m11B-all142a01.q1oz	krv03h12.q1	41	558	250	767	F	518	95.95 613
443	m11B-all142a01.q1oz	mse39371-828b06.q1oz	41	586	72	600	F	529	94.90 613
441	m11B-all142a01.q1oz	L25824P6007RG9.T0	41	586	849	302	C	546	94.14 613
425	m11B-all142a01.q1oz	L25824P6002RF4.T0	41	525	240	725	F	485	96.08 613
422	m11B-all142a01.q1oz	1nd86c01.b1	41	586	113	660	F	546	92.67 613
418	m11B-all142a01.q1oz	19866846414296	101	586	1	487	F	486	95.88 613
390	m11B-all142a01.q1oz	o1z10a08.q1	41	437	488	91	C	397	99.50 613
381	m11B-all142a01.q1oz	L25824P6004RB10.T0	41	438	431	32	C	398	98.74 613
381	m11B-all142a01.q1oz	G10P642886RH4.T0	142	586	752	307	C	445	95.73 613
376	m11B-all142a01.q1oz	L25824P6003RB10.T0	41	431	429	39	C	391	98.98 613
372	m11B-all142a01.q1oz	o1z07a02.b1	151	586	47	483	F	436	95.64 613
372	m11B-all142a01.q1oz	1tm42h09.q1	125	586	1025	565	C	462	93.94 613
370	m11B-all142a01.q1oz	o1z19d12.b1	41	532	446	927	F	492	93.50 613
369	m11B-all142a01.q1oz	G10P670226FP4.T0	41	426	336	722	F	386	98.45 613
369	m11B-all142a01.q1oz	L25824P6003RG6.T0	41	419	416	37	F	379	99.21 613
368	m11B-all142a01.q1oz	o1z08c08.b1	143	586	37	481	F	444	94.82 613
353	m11B-all142a01.q1oz	L25824P6004RB8.T0	41	400	392	32	C	360	99.44 613
351	m11B-all142a01.q1oz	L25824P674FF9.T0	100	586	750	264	C	487	91.58 613
350	m11B-all142a01.q1oz	G10P618880FE11.T0	173	586	721	307	C	414	95.17 613
346	m11B-all142a01.q1oz	L25824P6003RB8.T0	41	393	392	39	C	353	99.43 613
339	m11B-all142a01.q1oz	L25824P6006FG5.T0	41	396	386	29	C	356	98.60 613
331	m11B-all142a01.q1oz	L25824P6000PD2.T0	41	396	717	1073	F	356	97.75 613
324	m11B-all142a01.q1oz	L25824P6004RG6.T0	41	425	420	32	C	385	95.06 613
314	m11B-all142a01.q1oz	o1z13f06.b1	211	586	47	423	F	376	94.95 613
313	m11B-all142a01.q1oz	19866857420890	205	586	1	382	F	382	94.50 613
295	m11B-all142a01.q1oz	L25824P6005RB8.T0	41	396	44	403	C	356	94.66 613
249	m11B-all142a01.q1oz	o1z22f11.b1	272	586	702	387	C	315	93.02 613
236	m11B-all142a01.q1oz	mse39374-388q10.q1oz	41	286	413	659	F	246	99.19 613
233	m11B-all142a01.q1oz	19866857096035	277	586	5	313	F	310	92.58 613
228	m11B-all142a01.q1oz	G135P601378FP8.T0	41	278	246	8	C	228	99.16 613
207	m11B-all142a01.q1oz	o1z08d01.q1	310	586	617	340	C	277	92.06 613
183	m11B-all142a01.q1oz	19866839673813	41	241	593	796	F	201	98.01 613
153	m11B-all142a01.q1oz	19866835810955	41	202	163	1	C	162	98.77 613
108	m11B-all142a01.q1oz	19866822101368	222	363	84	225	F	142	92.25 613
98	m11B-all142a01.q1oz	19866822101368	221	363	488	632	F	143	90.21 613
94	m11B-all142a01.q1oz	19866822243206	214	358	209	62	C	145	88.97 613
72	m11B-all142a01.q1oz	19866838986547	218	326	27	134	F	109	89.91 613
64	m11B-all142a01.q1oz	vsa08h03_147e43.b1	221	326	218	325	F	106	87.74 613
51	m11B-all142a01.q1oz	19866822243206	215	304	493	404	C	90	86.67 613
46	m11B-all142a01.q1oz	vsa08h03_147e43.b1	314	363	547	596	F	50	98.00 613
33	m11B-all142a01.q1oz	19866838986547	314	358	278	322	F	45	91.11 613

Download the traces in either scf or fasta format by clicking on one of the buttons below.

[SCF](#) [FASTA](#)

Validation:

Validation:

⇒ Verification of sequence origin.

Validation:

- ⇒ Verification of sequence origin.
- ⇒ Confirmation of putative novel sequence.

Validation:

- ⇒ Verification of sequence origin.
- ⇒ Confirmation of putative novel sequence.
- ⇒ Contamination screening.

Validation:

- ⇒ Verification of sequence origin.
- ⇒ Confirmation of putative novel sequence.
- ⇒ Contamination screening.
- ⇒ Improving both correctness and length of assembly.

Finishing:

Finishing:

⇒ Find missing parts in the draft genome assemblies.

Finishing:

- ⇒ Find missing parts in the draft genome assemblies.
- ⇒ Closing difficult gaps using reads from other centres.

Add depth:

Add depth:

⇒ Reads from other centres/species.

Add depth:

- ⇒ Reads from other centres/species.
- ⇒ Localized comparative genomics.

Add depth:

- ⇒ Reads from other centres/species.
- ⇒ Localized comparative genomics.
- ⇒ Added confidence to SNPs and indels.

Add depth:

- ⇒ Reads from other centres/species.
- ⇒ Localized comparative genomics.
- ⇒ Added confidence to SNPs and indels.
- ⇒ Crude estimate of allele-frequency.

Add width:

Add width:

⇒ Extend the sequence in both directions

Add width:

- ⇒ Extend the sequence in both directions
- ⇒ e.g. Build-up an entire exon starting from an EST

Add width:

- ⇒ Extend the sequence in both directions
- ⇒ e.g. Build-up an entire exon starting from an EST
- ⇒ Local consensus around any given feature

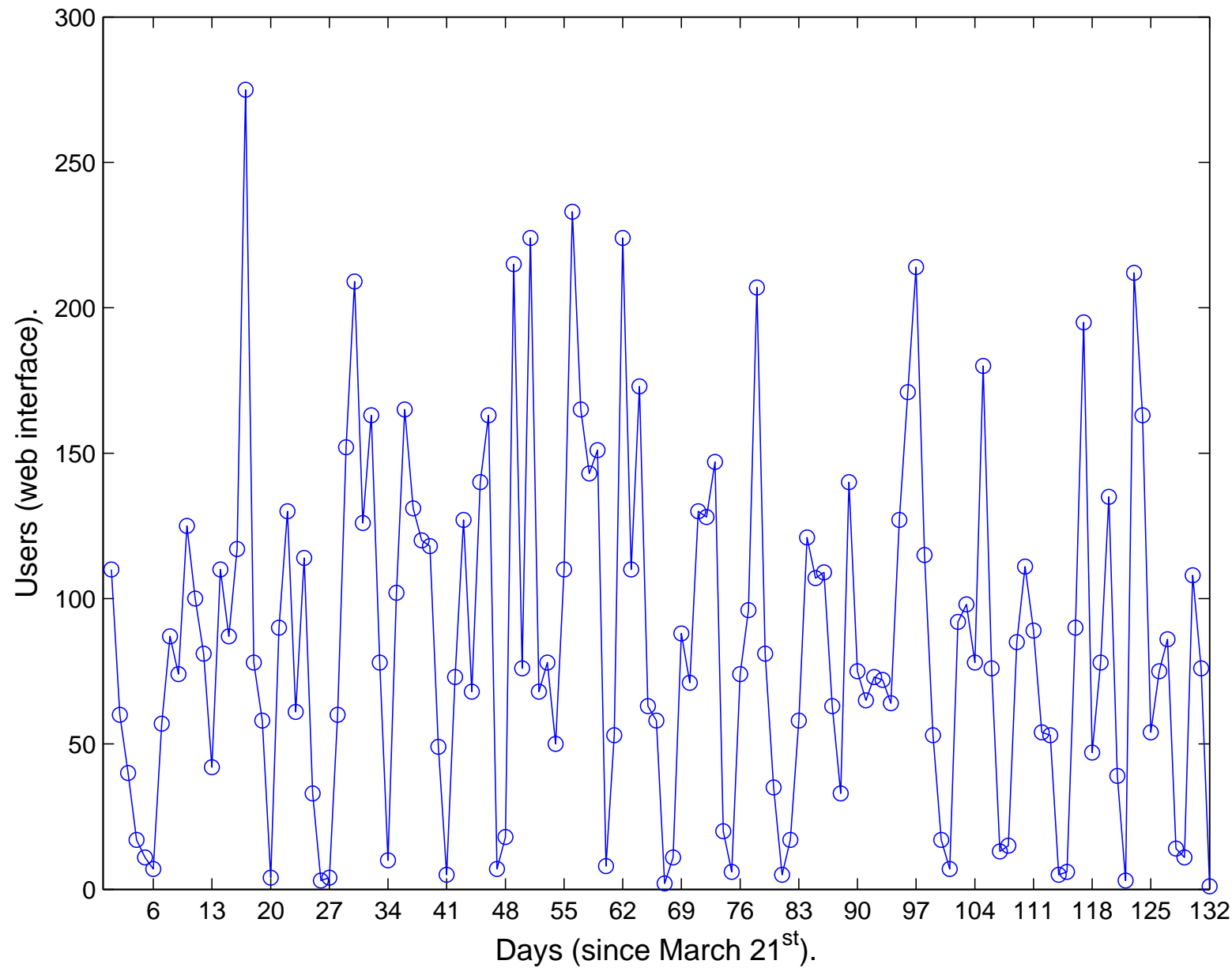
Add width:

- ⇒ Extend the sequence in both directions
- ⇒ e.g. Build-up an entire exon starting from an EST
- ⇒ Local consensus around any given feature
- ⇒ Recovery of novel sequence.

Add width:

- ⇒ Extend the sequence in both directions
- ⇒ e.g. Build-up an entire exon starting from an EST
- ⇒ Local consensus around any given feature
- ⇒ Recovery of novel sequence.
- ⇒ Checking computationally derived structural variations.

Any more? - please email aws@sanger.ac.uk.



<http://www.sanger.ac.uk/Software/analysis/SSAHA2/>